Deformable Graph Model for Tracking Epithelial Cell Sheets in Fluorescence Microscopy

Roger S. Zou and Carlo Tomasi

Abstract—**We propose a novel method for tracking cells that are connected through a visible network of membrane junctions. Tissues of this form are common in epithelial cell sheets and resemble planar graphs where each face corresponds to a cell. We leverage this structure and develop a method to track the entire tissue as a deformable graph. This coupled model in which vertices inform the optimal placement of edges and *vice versa* captures global relationships between tissue components and leads to accurate and robust cell tracking. We compare the performance of our method with that of four reference tracking algorithms on four data sets that present unique tracking challenges. Our method exhibits consistently superior performance in tracking all cells accurately over all image frames, and is robust over a wide range of image intensity and cell shape profiles. This may be an important tool for characterizing tissues of this type especially in the field of developmental biology where automated cell analysis can help elucidate the mechanisms behind controlled cell-shape changes.**

Index Terms—**Cell tracking, deformable graph model, Drosophila melanogaster, epithelial cell sheets, fluorescence microscopy, Scale-Invariant Feature Transform (SIFT) flow.**

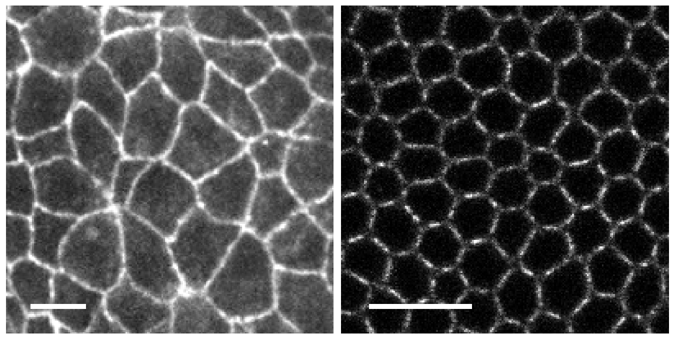


Fig. 1. Examples of epithelial cell sheets in the *Drosophila melanogaster* embryo. Left: 200×200 px image of amnioserosa cells during dorsal closure (image from [4]). Right: 200×200 px image of epithelium before dorsal fold formation (image from [5]). Both scale bars are 10 μm (px = pixel).

I. Introduction

QUANTITATIVE characterization of cell and tissue behavior from image data is important in many biological studies that aim to elucidate causal relationships between biochemical pathways and biophysical behavior. These investigations perturb tissues genetically or otherwise and quantify the resulting phenotypical changes.

Recent advances in live-imaging techniques have led to image data sets with high spatial and temporal resolution [1]. Fluorescence microscopy is frequently used for these studies to interrogate biological structures with high sensitivity and specificity [2]. Unfortunately, manual quantification of such data sets is very labor-intensive and frequently infeasible. Computer aided techniques are therefore necessary and usually involve the combination of tissue segmentation and tracking. However, these automated techniques face many challenges. First, such images frequently exhibit poor signal-to-noise ratios (SNR) due to background autofluorescence [3] and the low laser light intensities that are needed to limit photobleaching [2]. Second, biological structures commonly experience significant nonlinear deformations and large displacements between image frames. Third, different fluorescence labeling techniques may lead to diverse appearances, even for different samples of the same biological entity. Fourth, the image intensity profiles may gradually vary over the course of imaging due to changes in the biological structure and/or deviations in the intensities of the fluorophores.

Given these challenges, algorithms that incorporate prior information on the structure of the tracked object can potentially perform significantly better than general-purpose tracking methods, yet such algorithms should be sufficiently general for application to a wide range of biological systems. In this paper, we advance this philosophy by proposing a method for tracking a common biological structure: tissues composed of adjacent cells connected through a visible network of membrane junctions. This architecture is frequently found in epithelial cell sheets and is of special importance in the field of developmental biology, in which quantification of these tissues helps to elucidate the biochemical processes that underlie the tightly controlled physical transformations from a single cell into a complex, multicellular organism. Examples in which this architecture is observed can be found across phylogeny, ranging from *Drosophila melanogaster* (fruit fly) dorsal closure [4], ventral epithelium [6], dorsal folds [5], dorsal thorax [1], and ovary follicle cell epithelium [7] to *Danio rerio* (zebrafish) epiboly [1] to *Mus musculus* (mouse) anterior visceral endoderm [1] to *Arabidopsis thaliana* (flowering plant) shoot apical meristem [8] to *Caenorhabditis elegans* (roundworm) ventral enclosure [9]. Figure 1 shows two examples of such structures.

Manuscript received November 25, 2015; revised January 14, 2016; accepted January 21, 2016.

R.S. Zou and C. Tomasi are with the Department of Computer Science, Duke University, Durham, NC, 27708 USA. E-mail: roger.zou@duke.edu, tomasi@cs.duke.edu.

A. Related Work

The literature on cell tracking can be organized into two general methodologies. The first includes methods in which segmentation is first performed on every frame, followed by a separate matching process. Common approaches to segmentation involve the use of algorithms such as watershed [8], [10]–[12], image filtering [6], and thresholding techniques such as Otsu's method or locally adaptive thresholds [13], [14]. Matching schemes subsequently assign correspondences between segmented components across image frames using a variety of methods [8], [15], [16]. Segmentation is usually very efficient, but matching frequently requires complex and domain-specific strategies to account for ambiguous, non-bijective correspondences. Furthermore, segmentation is performed on each frame independently and does not utilize any high-level information about the underlying biological structure. The accumulation of tracking errors due to incorrect segmentation can also be problematic: While segmentation of one frame may be close to perfect, the number of tracking errors over the span of multiple frames may result in the tracker losing a significant number of cells.

The second methodology is a model evolution approach. An initial model is deformed to track the object of interest by using the results from the current frame as initialization to track in the next frame. Common examples include active contours [9], [17]–[21], level sets [22]–[24], and mean-shift tracking [25]. These methods often have the advantage of trivially obtaining segmentation and cell correspondences from the model, but are frequently more computationally expensive and may require domain-specific methods of handling object topology changes. In tracking groups of cells, these methods usually track cells individually and assume spatial independence between individual contours, which may or may not be valid depending on the specific application.

Both methodologies have also been previously applied to imagery closely related to the membrane networks we study, using either separate segmentation and matching [6], [8] or model evolution [9], [26] approaches.

B. Our Contributions

We distinguish our contributions into four categories. First, our proposed method tracks the entire tissue as a single structure, and utilizes this high-level information to improve tracking at the cellular level. Visual tracking data ranging from the tissue level to sub-cellular level is recorded. This distinguishes our method from other model evolution approaches that exclusively focus on independently tracking individual cells.

Second, our graph model completely parametrizes the behavior of the tissue with a pre-specified number of variables. This model is modular in that blocks with different models for vertices and edges of a graph can be easily swapped in and out within the same tracking framework, allowing for possible applications in multiple domains.

Third, our tracking method of optimizing a graph model is novel from an algorithmic perspective. Whereas frequently model evolution methods either track points or evolve curves, we track both in unison. A unique feature of our method is the tight coupling between vertices and edges: vertices guide the optimal placement of edges while edges guide the optimal placement of vertices for a solution that is optimal according to a more global criterion.

Fourth, although past contributions have offered methods for tracking biological structures of this type, they frequently focus on the biology and do not compare their algorithms to other methods in literature. We present a thorough analysis of a wide variety of methods through a consistent set of evaluation measures. Furthermore, the data sets used in our evaluations present unique tracking challenges that highlight strengths and weaknesses of different tracking methods. We hope this will be useful for biologists who need to make informed choices when selecting the best tracking algorithm for their applications. We also make our source code publicly available at http://github.com/rogerzou/cell-sheet-tracker.

II. Problem Statement and Techniques

Our method aims to track a graph that deforms in a video sequence, in which the first graph in the sequence is specified. This initial graph can be constructed from any segmentation, whether obtained by hand or by an algorithm. We assume that the state of the graph is Markovian, so that tracking only needs to consider two consecutive frames I and J. Given graph $G_I = (V_I, E_I)$ in image I, the objective is to determine its corresponding graph $G_J = (V_J, E_J)$ in image J. Although we implement our algorithm in $d = 2$ dimensions, the derivation below is general for $d \geq 2$. Furthermore, we assume that no new vertices or edges are created in frames after the first, although vertices and edges can merge and/or disappear. In practice, this means that our method can model events such as cell apoptosis, but not cell divisions or emergence of new cells.

Section II-A details the representation of our model. Section II-B presents an overview of our proposed tracking method. Sections II-C to II-E derive the main tracking equations. Section II-F explains how to incorporate SIFT flow [27] for improved tracking robustness and faster optimization convergence. Section II-G summarizes our method with pseudocode.

A. Graph Representation

We represent graph vertices as positions in the image. Biologically, they are placed on membrane branch-points where multiple membranes merge. A vertex $i \in V$ is represented by its position $\mathbf{v}_i$ and $(i, j) \in E$ designates positions $\mathbf{v}_i$ and $\mathbf{v}_j$ to be connected by an edge.

We represent edge geometry with B-splines [28] whose endpoints are constrained to vertices. Biologically, edges trace the visible membrane junctions that link two adjacent cells. An open spline in d dimensions between vertex positions $\mathbf{v}_i$ and $\mathbf{v}_j$ with $k + 2$ control points is a curve of parametric form

$$\boldsymbol{\gamma}(s, \mathbf{q}_{ij}) = P(s)\,\mathbf{q}_{ij} \quad \text{for} \quad s \in [0, 1]\,, \tag{1}$$

where $\mathbf{q}_{ij}$ is a vector of dimension $m = d(k+2)$ that collects the control points and $P(s) \in \mathbb{R}^{d\times m}$ is a polynomial matrix function of the curve parameter s. The first and last control

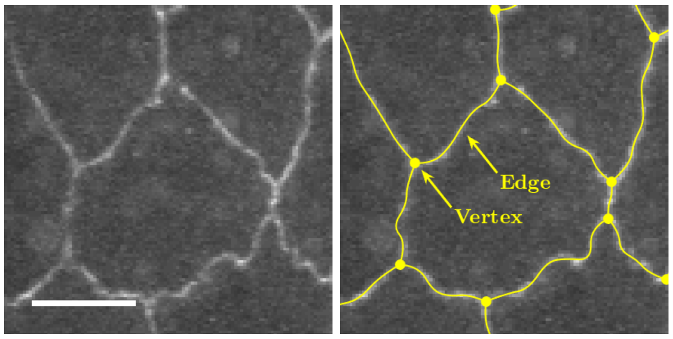


Fig. 2. Graph representation. Left: 108 × 108 px image of a cell on the dorsal opening of a *Drosophila* embryo. Right: same image with our graph representation overlaid. Filled circles represent vertices, curves represent edges, and the graph face represents a cell. The scale bar is 10 μm. [Color picture online.]

points (the *endpoints*) of $\mathbf{q}_{ij}$ are constrained to coincide with the vertex positions $\mathbf{v}_i$ and $\mathbf{v}_j$:

$$\mathbf{q}_{ij} = \left[\mathbf{v}_i^\top, \boldsymbol{\xi}_{ij}^\top, \mathbf{v}_j^\top\right]^\top \in Q(i,j)\,, \tag{2}$$

where $\boldsymbol{\xi}_{ij}$ is the vector of *interior* control points of dimension $r = dk$ and $Q(i,j)$ is the space of control points that satisfy these constraints. Thus, the graph is completely parametrized with d numbers for each vertex and r numbers for each edge. The number of control points in each spline is determined by a scalar parameter ρ that specifies the approximate distance between control points for all edges. Figure 2 illustrates our graph representation on a sample cell.

B. Tracking Algorithm

This representation is naturally amenable to measures of dissimilarity (what we will henceforth call 'cost') between graphs embedded in images. Tracking a graph between images I and J entails deforming G_I in I to some G_J in J such that the cost C_G between G_I and G_J is minimized. We construct C_G from individual vertex and edge costs.

1) Vertex cost: We define the vertex cost using a least-squares dissimilarity measure [29]; the cost c_i of placing a vertex at position $\mathbf{v}_i \in J$ given that the same vertex was at position $\mathbf{u}_i \in I$ is

$$c_i = \int_{\mathbb{R}^d} [r_i(\mathbf{x})]^2 \, w_v(\mathbf{x})\, d\mathbf{x}\,, \tag{3}$$

where

$$r_i(\mathbf{x}) = I(\mathbf{x} + \mathbf{u}_i) - J(\mathbf{x} + \mathbf{v}_i) \tag{4}$$

is the *vertex residual* and $w_v(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^+$ is a truncated zero-mean Gaussian function that localizes the integral to a finite interval of length ω in each dimension. This is the first of many definitions of *costs* and *residuals* in this paper; their explicit dependence on the graph model is not shown, unless needed for emphasis (For example, in this case c_i and $r_i(\mathbf{x})$ are unambiguously also functions of $\mathbf{u}_i$ and $\mathbf{v}_i$).

2) Edge cost: We define the edge cost using a similar least-squares dissimilarity measure extended to curves [28]. For a spline $\boldsymbol{\gamma}(s, \mathbf{q}_{ij})$ between vertex positions $\mathbf{v}_i$ and $\mathbf{v}_j$, the edge cost of assigning $\mathbf{q}_{ij}$ to the spline in J given that the same spline had control points $\mathbf{p}_{ij}$ in I is

$$c_{ij} = \int_0^1 \left\{ \int_{\mathbb{R}^{d-1}} [r_{ij}(s,\mathbf{t})]^2 \, w_e(\mathbf{t})\, d\mathbf{t} \right\} \sigma(s)\, ds\,, \tag{5}$$

where

$$r_{ij}(s,\mathbf{t}) = I(\mathbf{y}(s,\mathbf{t},\mathbf{p}_{ij})) - J(\mathbf{y}(s,\mathbf{t},\mathbf{q}_{ij})) \tag{6}$$

is the *edge residual* and

$$\mathbf{y}(s,\mathbf{t},\mathbf{q}_{ij}) = \boldsymbol{\gamma}(s,\mathbf{q}_{ij}) + F(s,\mathbf{q}_{ij})\,\mathbf{t} \tag{7}$$

is a function that specifies an image sampling point for some s and $\mathbf{t}$. The parameter $s \in [0,1]$ varies along the tangent vector and $\mathbf{t} \in \mathbb{R}^{d-1}$ varies along the remaining $d-1$ Frenet vectors of $\boldsymbol{\gamma}$. These remaining vectors compose the columns of $F(s,\mathbf{q}_{ij})$, and column F_k can be written as a linear function

$$F_k = N_k(s)\,\mathbf{q}_{ij} \tag{8}$$

of the control points. The truncated zero-mean Gaussian function $w_e : \mathbb{R}^{d-1} \to \mathbb{R}^+$ localizes the inner integral to an interval of length ℓ in each dimension and $\sigma(s)$ is the speed at which the point $\boldsymbol{\gamma}(\cdot, s)$ travels on $\boldsymbol{\gamma}$.

3) Graph cost: Because the problem of estimating the edge path between vertex positions $\mathbf{v}_i$ and $\mathbf{v}_j$ is independent of the problem of finding the edge path between any other pair of vertex positions, the independent variables of the graph tracking problem are the vertex positions $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of the graph in J, and the optimal spline control points $\mathbf{q}_{ij}^*$ for $(i,j) \in E$ are in turn functions of the vertex positions. The optimization problem is therefore

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} C_G(\mathbf{z})\,, \tag{9}$$

where

$$C_G(\mathbf{z}) = \alpha \sum_{i \in V} c_i + (1-\alpha) \sum_{(i,j)\in E} c_{ij}(\mathbf{q}_{ij}^*) \tag{10}$$

is the *graph cost* function. The convex weight $\alpha \in [0,1]$ weighs the contributions of vertex and edge costs,

$$\mathbf{z} = \left[\mathbf{v}_1^\top \quad \ldots \quad \mathbf{v}_n^\top\right]^\top \tag{11}$$

is a column vector that collects all the vertex positions, and

$$\mathbf{q}_{ij}^* = \arg \min_{\mathbf{q}_{ij} \in Q(i,j)} c_{ij}(\mathbf{q}_{ij}) \tag{12}$$

is the vector of spline control points that minimizes c_{ij} over the space $Q(i,j)$ (introduced in Equation 2).

The minimization procedure we use for both Equations 9 and 12 is the Gauss-Newton algorithm [30], which requires the gradient and first-order Hessian approximation (which we will henceforth call the 'Gramian') of their respective objective functions at each iteration. Section II-C derives the derivatives necessary to solve the *edge* problem in Equation 12. Sections II-D and II-E derive the derivatives necessary to solve the *graph* problem in Equation 9.

C. Edge-Optimization Derivatives

Determining the optimal control points $\mathbf{q}_{ij}^*$ in Equation 12 with the Gauss-Newton algorithm requires both the gradient $\nabla_{\mathbf{q}}\, c_{ij}$ and Gramian $\mathcal{G}_{\mathbf{q}}\, c_{ij}$ of the edge cost c_{ij} in Equation 5.

To obtain these quantities, we begin by computing the gradient of the residual in Equation 6 with respect to $\mathbf{q}_{ij}$,

$$\nabla_{\mathbf{q}} r_{ij}(s, \mathbf{t}) = -\left[\frac{\partial \mathbf{y}(s, \mathbf{t}, \mathbf{q}_{ij})}{\partial \mathbf{q}_{ij}^\top}\right]^\top \nabla J\left(\mathbf{y}(s, \mathbf{t}, \mathbf{q}_{ij})\right) \,, \quad (13)$$

where $\nabla J(\mathbf{y})$ is the gradient vector of image J at $\mathbf{y}$. In Equation 13, the Jacobian of $\mathbf{y}$ with respect to $\mathbf{q}_{ij}$,

$$\frac{\partial \mathbf{y}(s, \mathbf{t}, \mathbf{q}_{ij})}{\partial \mathbf{q}_{ij}^\top} = P(s) + \sum_{k=1}^{d-1} N_k(s)\, t_k \,, \quad (14)$$

follows from Equations 1, 7, and 8. We now have the necessary components to construct the gradient

$$\nabla_{\mathbf{q}} c_{ij} = 2 \iint r_{ij} \left[\nabla_{\mathbf{q}} r_{ij}\right]\, w_e(\mathbf{t})\, d\mathbf{t}\, \sigma(s)\, ds \quad (15)$$

and the Gramian

$$\mathcal{G}_{\mathbf{q}} c_{ij} = 2 \iint \left[\nabla_{\mathbf{q}} r_{ij}\right] \left[\nabla_{\mathbf{q}} r_{ij}\right]^\top\, w_e(\mathbf{t})\, d\mathbf{t}\, \sigma(s)\, ds \quad (16)$$

of c_{ij}, where dependence of r_{ij} and $\nabla_{\mathbf{q}} r_{ij}$ on s, $\mathbf{t}$, and $\mathbf{q}_{ij}$ is left implicit and integration limits are as in Equation 5.

D. Effect of Vertex Displacement on the Optimal Spline

Determining the optimal graph parameters $\mathbf{z}^*$ in Equation 9 using the Gauss-Newton algorithm entails iteratively shifting the collection of vertices $\mathbf{z}$ to $\mathbf{z}^*$ using the gradient $\nabla_{\mathbf{z}} C_G$ and Gramian $\mathcal{G}_{\mathbf{z}} C_G$ of the graph cost C_G in Equation 10. In this equation, the optimal spline parameters $\mathbf{q}_{ij}^*$, for $\{j \mid (i,j) \in E\}$, are functions of the changing vertex positions, so to correctly compute the derivatives of the graph cost we must consider the effects of displacing a spline's endpoints (components of $\mathbf{z}$) on the displacements of its interior control points $\boldsymbol{\xi}$ in order to maintain edge optimality.

To this end, suppose that for some spline $\boldsymbol{\gamma}(s, \mathbf{q}_{ij})$ with endpoints $\mathbf{v}_i$ and $\mathbf{v}_j$, we shift $\mathbf{v}_i$ by an infinitesimal amount. Then a necessary condition for the spline to be optimal is that the gradient of c_{ij} with respect to the *internal* control points $\boldsymbol{\xi}_{ij}$ equals the zero vector:

$$\frac{\partial c_{ij}\left(\mathbf{v}_i, \boldsymbol{\xi}_{ij}(\mathbf{v}_i, \mathbf{v}_j), \mathbf{v}_j\right)}{\partial \boldsymbol{\xi}_{ij}} = \mathbf{0} \,, \quad (17)$$

where c_{ij} is written explicitly as a function of all three components of $\mathbf{q}_{ij}$, and $\boldsymbol{\xi}_{ij}$ is written explicitly as a function of $\mathbf{v}_i$ and $\mathbf{v}_j$ for emphasis. Then the Jacobian of Equation 17 with respect of $\mathbf{v}_i$ is

$$\frac{\partial}{\partial \mathbf{v}_i^\top}\left[\frac{\partial c_{ij}}{\partial \boldsymbol{\xi}_{ij}}\right] = \frac{\partial^2 c_{ij}}{\partial \boldsymbol{\xi}_{ij}\, \partial \mathbf{v}_i^\top} + \frac{\partial^2 c_{ij}}{\partial \boldsymbol{\xi}_{ij}\, \partial \boldsymbol{\xi}_{ij}^\top} \frac{\partial \boldsymbol{\xi}_{ij}}{\partial \mathbf{v}_i^\top} = 0 \,, \quad (18)$$

and the implicit function theorem entails

$$\frac{\partial \boldsymbol{\xi}_{ij}^*}{\partial \mathbf{v}_i^\top} = -\left[\frac{\partial^2 c_{ij}}{\partial \boldsymbol{\xi}_{ij}\, \partial \boldsymbol{\xi}_{ij}^\top}\right]^{-1}\Bigg|_{\boldsymbol{\xi}_{ij} = \boldsymbol{\xi}_{ij}^*} \left[\frac{\partial^2 c_{ij}}{\partial \boldsymbol{\xi}_{ij}\, \partial \mathbf{v}_i^\top}\right]\Bigg|_{\boldsymbol{\xi}_{ij} = \boldsymbol{\xi}_{ij}^*} \quad (19)$$

for the optimal solution (assuming the first factor in the right-hand side is invertible). Both parts of Equation 19 can be approximated with sub-matrices of $\mathcal{G}_{\mathbf{q}}\, c_{ij}$ in Equation 16. We can now quantify how the optimal interior control points change with respect to either endpoint ($\mathbf{v}_i$ shown):

$$\frac{\partial \mathbf{q}_{ij}^*}{\partial \mathbf{v}_i^\top} = \left[\mathbf{I}_d \,, \left[\frac{\partial \boldsymbol{\xi}_{ij}^*}{\partial \mathbf{v}_i^\top}\right]^\top , \, \mathbf{0}_d\right]^\top \,, \quad (20)$$

where $\mathbf{I}_d$ and $\mathbf{0}_d$ are the $d \times d$ identity and zero matrices, respectively. As we shall see in Section II-E, the Jacobian in Equation 20 is a crucial component of the graph cost derivatives.

E. Graph-Optimization Derivatives

To derive expressions for $\nabla_{\mathbf{z}} C_G$ and $\mathcal{G}_{\mathbf{z}} C_G$, we need the gradient of the vertex cost in Equation 3 with respect to $\mathbf{v}_i$,

$$\nabla_i\, c_i = 2 \int r_i(\mathbf{x}) \left[\nabla_i r_i(\mathbf{x})\right] w_v(\mathbf{x})\, d\mathbf{x} \,. \quad (21)$$

The gradient with respect to $\mathbf{v}_i$ of the residual $r_i(\mathbf{x})$ (Equation 4) is

$$\nabla_i\, r_i(\mathbf{x}) = -\nabla J(\mathbf{x} + \mathbf{v}_i) \,, \quad (22)$$

the image gradient of J evaluated at $\mathbf{x} + \mathbf{v}_i$. Similarly, the gradient of the edge cost in Equation 5 for an optimal spline with respect to $\mathbf{v}_i$ (assuming vertex $\mathbf{v}_j$ is unchanged) is

$$\nabla_i\, c_{ij} = 2 \iint r_{ij} \left[\nabla_i r_{ij}\right] w_e(\mathbf{t})\, d\mathbf{t}\, \sigma(s)\, ds \,, \quad (23)$$

where $r_{ij} = r_{ij}(s, \mathbf{t}, \mathbf{q}_{ij})$ is the edge residual defined in Equation 6. Its gradient with respect to $\mathbf{v}_i$ is

$$\nabla_i\, r_{ij} = -\left[\frac{\partial \mathbf{q}_{ij}^*}{\partial \mathbf{v}_i^\top}\right]^\top \left[\frac{\partial \mathbf{y}}{\partial \mathbf{q}_{ij}^\top}\right]^\top\Bigg|_{\mathbf{q}_{ij} = \mathbf{q}_{ij}^*} \nabla J(\mathbf{y}) \,, \quad (24)$$

where $\mathbf{y} = \mathbf{y}(s, \mathbf{t}, \mathbf{q}_{ij})$. The first factor in the right-hand side of Equation 24 is the Jacobian defined in Equation 20, the second factor is defined in Equation 14, and the third factor is the image gradient evaluated at $\mathbf{y}$. We then have the components for the gradient of the graph cost in Equation 10 with respect to $\mathbf{v}_i$,

$$\nabla_i\, C_G = \alpha \left[\nabla_i\, c_i\right] + (1 - \alpha) \sum_{k \in \mathcal{N}(i)} \left[\nabla_i\, c_{ik}(\mathbf{q}_{ik}^*)\right] \,, \quad (25)$$

where $\mathcal{N}(i)$ is the set of vertices corresponding to vertex positions $\mathbf{v}_k$ that are connected to $\mathbf{v}_i$ by an edge with optimal control points $\mathbf{q}_{ik}^*$. To construct the Gramian of the graph cost, we first obtain the Gramian of the vertex cost,

$$\mathcal{G}_{ii}\, c_i = 2 \int \left[\nabla_i r_i(\mathbf{x})\right] \left[\nabla_i r_i(\mathbf{x})\right]^\top w_v(\mathbf{x})\, d\mathbf{x} \,, \quad (26)$$

the Gramian of the edge cost with respect to $\mathbf{v}_i$,

$$\mathcal{G}_{ii}\, c_{ij} = 2 \iint \left[\nabla_i r_{ij}\right] \left[\nabla_i r_{ij}\right]^\top w_e(\mathbf{t})\, d\mathbf{t}\, \sigma(s)\, ds \,, \quad (27)$$

and the Gramian of the edge cost with respect to $\mathbf{v}_i$ and $\mathbf{v}_j$,

$$\mathcal{G}_{ij}\, c_{ij} = 2 \iint \left[\nabla_j r_{ij}\right] \left[\nabla_i r_{ij}\right]^\top w_e(\mathbf{t})\, d\mathbf{t}\, \sigma(s)\, ds \,. \quad (28)$$

Equations 26 and 27 yield the diagonal components of the graph cost Gramian with respect to $\mathbf{v}_i$,

$$\mathcal{G}_{ii}\, C_G = \alpha\, [\mathcal{G}_{ii}\, c_i] + (1-\alpha) \sum_{k \in \mathcal{N}(i)} [\mathcal{G}_{ii}\, c_{ik}(\mathbf{q}^*_{ik})] \;, \tag{29}$$

and Equation 28 yields the off-diagonal components with respect to $\mathbf{v}_i$ and $\mathbf{v}_j$ for $(i,j) \in E$ (otherwise, $\mathcal{G}_{ij} c_{ij} = 0$),

$$\mathcal{G}_{ij}\, C_G = (1-\alpha) \left[\mathcal{G}_{ij}\, c_{ij}(\mathbf{q}^*_{ij})\right] \;. \tag{30}$$

The gradient in Equation 25 is the i^{th} component of the graph cost gradient

$$\nabla_{\mathbf{z}} C_G = \left[[\nabla_1 C_G]^\top \ldots [\nabla_n C_G]^\top \right]^\top \tag{31}$$

while the terms in Equations 29 and 30 are the $(i,i)^{th}$ and $(i,j)^{th}$ components of the graph-cost Gramian

$$\mathcal{G}_{\mathbf{z}} C_G = \begin{bmatrix} \mathcal{G}_{11} C_G & \cdots & \mathcal{G}_{1n} C_G \\ \vdots & & \vdots \\ \mathcal{G}_{n1} C_G & \cdots & \mathcal{G}_{nn} C_G \end{bmatrix} \;. \tag{32}$$

F. SIFT Flow for Vertex Initialization

SIFT flow [27] is designed to robustly align images that differ through complex spatial distortions, and Lee *et al.* [9] used this method to improve active contour tracking. We incorporate a similar idea by using the final vertex positions $\mathbf{z}^*$ of the current frame, displaced with SIFT flow, as initialization to track the next frame. Better initialization helps to avoid local minima and ensure faster convergence during optimization. Compared to simply using $\mathbf{z}^*$ of the current frame as initialization for the next frame, in practice this addition significantly reduces the number of Newton iterations and improves tracking robustness.

G. Method Summary

Algorithm 1 summarizes the function GRAPHOPT for computing the initial graph G_0 and the graph tracking results $G_1, \ldots, G_N$ from the image stream $I_0, \ldots, I_N$. In line 2, the initial graph G_0 can be constructed from any segmentation of the initial frame (obtained manually or automatically). Each iteration of the **for** loop in lines 3-11 finds the optimal graph for each image, given the optimal graph from the previous image. UPDATEGRAPHTOPOLOGY in line 4 merges any vertices whose distances are below a fixed threshold of 2 pixels and handles the resulting graph topology changes. SIFTFLOW in line 5 uses SIFT flow to compute the initial graph vertex positions for graph G_i in image I_i, and the **repeat** loop in lines 6-10 computes a solution to Equation 9 for G_i using the Gauss-Newton algorithm. In so doing, the EDGEOPT function in line 9 computes a solution to Equation 12 for every edge of the graph so as to maintain edge optimality as the vertex positions move. Algorithm 2 details the internals of EDGEOPT. The **repeat** loop in lines 4-7 solves Equation 12 for a particular edge e_k using the Gauss-Newton algorithm.

Algorithm 1 Graph optimization

```
1:  function GRAPHOPT($I_0, \ldots, I_N$)
2:      compute initial graph $G_0 = (V_0, E_0)$ from $I_0$
3:      for $i = 1, \ldots, N$ do
4:          $G_i \leftarrow$ UPDATEGRAPHTOPOLOGY($G_{i-1}$)
5:          $G_i \leftarrow$ SIFTFLOW($G_i, I_{i-1}, I_i$)
6:          repeat
7:              compute gradient $\nabla_{\mathbf{z}} C_G$ and Gramian $\mathcal{G}_{\mathbf{z}} C_G$
8:              compute and take Newton step $\Delta\mathbf{z}$ on $G_i$
9:              $G_i \leftarrow$ EDGEOPT($G_i, I_{i-1}, I_i$)
10:         until convergence
11:     end for
12:     return $G_0, \ldots, G_N$
13: end function
```

Algorithm 2 Internal edge optimization

```
1:  function EDGEOPT($G = (V, E), I_1, I_2$)
2:      for $k = 1, \ldots, |E|$ do
3:          retrieve edge $e_k = (i, j) \in E$
4:          repeat
5:              compute gradient $\nabla_{\mathbf{q}} c_{ij}$ and Gramian $\mathcal{G}_{\mathbf{q}} c_{ij}$
6:              compute and take Newton step $\Delta\mathbf{q}$ on $e_k$
7:          until convergence
8:          update $E$ with updated $e_k$
9:      end for
10:     return $G$
11: end function
```

III. EXPERIMENTAL RESULTS

We compared the performance of our tracking algorithm to that of other published methods. Section III-A introduces the four evaluation data sets. Section III-B describes the four published methods we compare to. Section III-C describes the two measures used to quantify tracking performance. Section III-D presents the parameters used for all evaluation experiments and Section III-E presents the results of these experiments. Section III-F presents a sensitivity analysis of our algorithm parameters and discusses heuristics for selecting good values for them. Section III-G provides more insight into the importance of edge contributions for better tracking. Section III-H presents the computation times for all algorithms. Section III-I discusses limitations of our approach.

A. Image Data Sets

We evaluate trackers on four data sets. They all span 101 frames and exhibit significantly different SNR, membrane shapes, and image intensity profiles. These differences pose unique challenges to different tracking algorithms (Figure 3).

Three data sets (DDC1, DDC4, DNC1) consist of time-lapse images of dorsal closure in the *D. melanogaster* embryo. This two-hour process occurs late into embryogenesis, and consists of approximately 200 cells that constrict along the anterior-posterior axis to form a seamless epithelium. These data sets

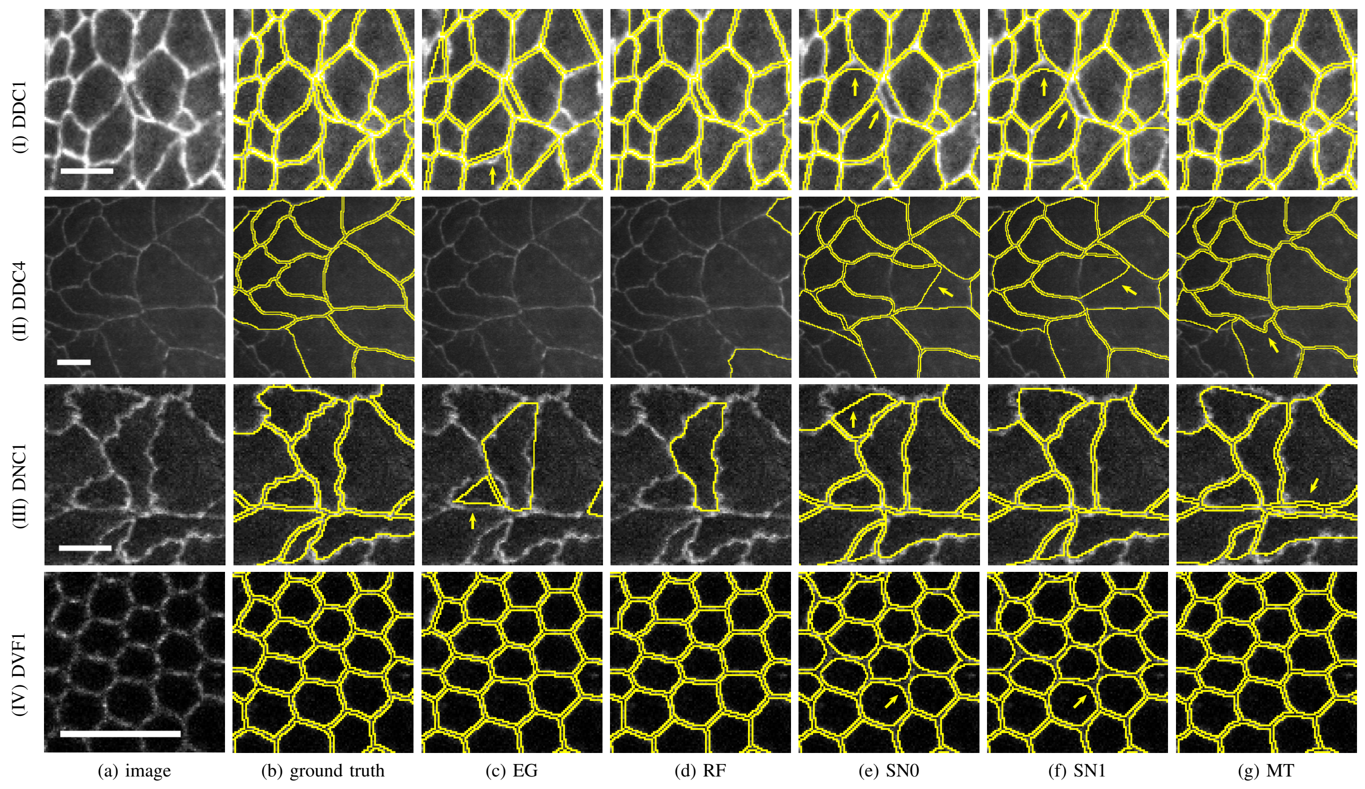


Fig. 3. Sample tracking results for different algorithms and data sets, all at the last frame (frame 101). A region of each data set is selected to highlight certain features in tracking results. Each overlaid, closed loop represents a tracked cell. Certain tracking errors are emphasized with arrows. Columns: (a) original image, (b) hand-labeled ground truth, (c-g) results from different tracking algorithms. MT refers to our proposed method. Rows: (I) 101 × 101 px image crop of DDC1; (II) 180 × 180 px image crop of DDC4; (III) 101 × 101 px image crop of DNC1; (IV) 101 × 101 px image crop of DVF1. All scale bars are 10 μm. [Color picture online.]

were acquired from live embryos using a 40x water immersion objective on a Zeiss Axioplan confocal microscope, and were obtained from the authors of [4]. DDC1 shows a wild-type embryo *during* dorsal closure, containing 82 visible cells imaged at 10 seconds per frame (272×483 px, 0.352 μm/px resolution, px=pixels). DDC4 shows a wild-type embryo *before* dorsal closure, containing 42 visible cells imaged at 30 seconds per frame (672 × 512 px, 0.305 μm/px resolution). DNC1 shows a mutant *nompC* embryo with a mechanically gated ion channel (MGC) deficiency [31] during dorsal closure, containing 25 visible cells imaged at 15 seconds per frame (512 × 272 px, 0.352 μm/px resolution).

The fourth data set (DVF1) consists of images, along the depth dimension, of the *D. melanogaster* dorsal epithelium before dorsal fold formation in a wild-type embryo. Dorsal fold formation is a 30-minute process occurring three hours into embryogenesis that produces two epithelial folds. This data set was acquired from a heat-methanol fixed embryo using a 63x multi-immersion objective on a Leica SP5 spectral confocal microscope, and was obtained from the authors of a previous study [5]. It shows 98 visible cells imaged spatially (300 × 300 px, 0.160 μm/px resolution) at a depth resolution of 0.126 μm between consecutive slices.

B. Other Algorithms for Comparison

We compared our algorithm to four others published in the literature. The EDGE method in Gelbart *et al.* [6] (which we will refer to as EG) and the conditional random field method in Chakraborty *et al.* [8] (which we will refer to as RF) both segment each frame separately before matching segmented regions. In contrast, the traditional snakes used as a baseline [32] (which we will refer to as SN0), the snakes with SIFT flow method in Lee *et al.* [9] (which we will refer to as SN1), and our method (which we will refer to as MT), all follow the model evolution approach of deforming an initial model over all frames.

To reconcile these differences between tracking methods in a fair evaluation, we ensured that other methods utilize at least as much of the same prior information as our own. We accomplished this task by supplying each method with the same cell segmentations. Since both EG and RF suggest a segmentation method, we used the one that performed better for each data set. We experimented with the watershed algorithm in MATLAB's image processing toolbox to implement the watershed segmentation [33] suggested by RF as well as a filtering/thresholding method suggested by EG, taken directly from their source code. Since we found that watershed consistently performed better on all data sets, we used MATLAB's watershed algorithm to generate segmentations for all data sets. A detail that affects the quality of watershed is an initial H-minima transform to suppress shallow minima, which is controlled by the `hminima` parameter that specifies the suppression height threshold. A low `hminima` results in over-segmentation and a high `hminima` results in under-segmentation. The optimal values of `hminima` were found through a manual, exhaustive search over a reasonable parameter range to yield the visibly best segmentation over all frames, as judged by a biologist. EG and RF used this segmentation for their matching steps, while MT, SN0, and SN1 used the first frame of it to automatically construct their initial models.

We also performed light Gaussian filtering on each data set before tracking to smooth out disruptive noise. Preprocessing methods tailored to enhancing cell membranes could be another option [34], [35].

We obtained source code for EG and RF from their authors, and implemented SN0 according to a classic recipe [32]. We were unable to obtain code for SN1, so we implemented that method ourselves to the best of our ability following its description [9]. We did not implement the non-intersecting force specified in that paper because our data sets did not exhibit the problem of intersecting contours.

C. Ground Truth and Tracking Evaluation Methods

For ground truth, a human expert segmented, every 10 frames, all cells that remain visible for the entire 101 frames of each data set. Given a cell C_i in the initial frame, let Y_i be the true region corresponding to C_i in a human-segmented frame, and let X_i be the region the tracker returns in that frame for cell C_i. The *region* of a cell is defined to be the set of image pixels inside the cell. We say that X_i *corresponds* to Y_i if the area $|Y_i \cap X_i|$ of the overlap region of Y_i and X_i is not empty and Y_i overlaps more with X_i than with any other X_j in that frame. If X_i and X_j overlap with Y_i the same amount, then X_i, where $i < j$, is selected. Otherwise, C_i is *lost*, either because X_i is missing (*i.e.* the tracker lost the cell) or because Y_i overlaps more with some other X_j in that frame (*i.e.* the tracker incorrectly swapped correspondences). We then evaluate tracking performance in two distinct ways:

- *δ-region error* evaluates an algorithm's ability to accurately delineate the region that corresponds to each cell.
- *% lost cells* evaluates an algorithm's ability to correctly track as many cells as possible over multiple frames.

Together, these two measures evaluate an algorithm's ability to correctly track the greatest number of cells with the greatest region accuracy for each cell. Better algorithms have lower values on both measures.

1) δ-region error ($\mathcal{E}_\delta$): This quantity measures the error in region overlap between ground truth cell regions and their corresponding tracked cell regions. However, region overlap is an uncertain quantity because the visible membranes junctions that delineate cell boundaries are generally more than 1 pixel wide. Thus, the true cell region exists in a space of possible regions, and the ground truth is just one sample from that space. We then formulate the error measure so that any region whose boundary is everywhere within some $\delta > 0$ distance from the true boundary is deemed to have no error. More specifically, let set X_i represent the tracked region of cell C_i and $Y_i(x)$ represent its corresponding ground truth region, dilated ($x > 0$) or eroded ($x < 0$) by x pixels. $Y_i(0)$ is the original ground truth region, $Y_i(\delta)$ indicates the $Y_i(0)$ region

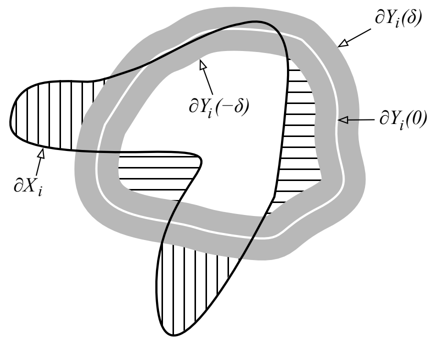


Fig. 4. Definition of δ-region error. The black curve is the outer boundary of the tracked region X_i of C_i. The white curve is the outer boundary of the ground-truth region $Y_i(0)$, and the outer and inner edge of the grey band are the outer boundaries of the dilated and eroded versions $Y_i(\delta)$ and $Y_i(-\delta)$ of $Y_i(0)$. Vertical hatches cover the false positive regions $X_i \cap \neg Y_i(\delta)$ and horizontal hatches cover the false negative regions $\neg X_i \cap Y_i(-\delta)$.

dilated by δ pixels, and $Y_i(-\delta)$ indicates the $Y_i(0)$ region eroded by δ pixels. We then define the *outside region* for C_i to be

$$Z_i(\delta) = \big[X_i \cap \neg Y_i(\delta)\big] \cup \big[\neg X_i \cap Y_i(-\delta)\big]\,, \tag{33}$$

the union of both false positive and false negative pixels that do not lie within a 2δ-wide band around the ground-truth region boundary. Figure 4 illustrates the definition of $Z_i(\delta)$. We can then define the *δ-region error* over all tracked cells in a particular frame to be

$$\mathcal{E}_\delta = \frac{|\bigcup_i Z_i(\delta)|}{|\bigcup_i Y_i(0)|} \times 100\,, \tag{34}$$

the size of the union of all outside regions (both false positives and false negatives, as compared with the reference regions), divided by the size of the union of all ground truth (or reference) regions. This measure is not a percentage, and it is possible in principle for $\mathcal{E}_\delta$ to be greater than 100. A biologist determined $\delta = 3$ pixels to be reasonable for all four data sets.

2) % lost cells ($\mathcal{L}_c$): This quantity measures the percent of all cells that an algorithm has lost at a particular time point. Formally, let S represent the set of ground truth cells and T represent the set of all ground truth cells that correspond to tracked cells. Since $T \subseteq S$ and we do not evaluate the introduction of new cells (*i.e.* a cell that is included after the first frame is excluded from consideration), it must be the case that $|T| \leq |S|$. Then the percent of lost cells is given by

$$\mathcal{L}_c = \frac{|S| - |T|}{|S|} \times 100\,. \tag{35}$$

D. Experimental Parameter Settings

Our tracking algorithm relies on four parameters: ω, the pixel width of the $\mathbb{R}^d$ integration interval for vertex cost in Equation 3; ℓ, the pixel width of the $\mathbb{R}^{d-1}$ integration interval for edge cost in Equation 5; α, the convex weight on the vertex and edge contributions in Equation 10; and ρ, the approximate pixel spacing between interior control points. For our evaluation, we chose these parameters to be $\omega = 21$,

TABLE I
OPTIMAL PARAMETERS FOUND WITHIN RANGE [MIN, MAX] FOR EACH DATA SET (COLUMNS) AND REFERENCE METHOD (ROW BLOCKS).

	param	min	DDC1	DDC4	DNC1	DVF1	max
EG	`tac`	0	0.6000	0.6058	0.6000	0.6000	1
	`tlb`	2	5	8	5	5	10
	`tcd`	1	10.000	3.3080	10.000	10.000	20
RF	λ_2	0.01	0.0667	0.0752	0.0268	0.0667	10
	λ_1	10	100.00	44.210	18.701	100.00	100
	w	0.25	0.7500	0.4043	0.3517	0.7500	0.75
	γ	0.01	0.1250	0.6414	0.3847	0.1250	1.0
SN0	α	0.1	0.3140	1.3558	0.4140	1.0293	2
	β	0.1	1.6554	1.0926	0.1066	1.1056	2
	γ	0.1	1.1255	1.8137	1.1417	0.8675	2
	κ	0.1	0.1202	0.1013	0.1530	0.1566	0.2
SN1	α	0.1	0.1467	1.4341	0.1456	1.0293	2
	β	0.1	0.5042	1.3773	0.4618	1.1056	2
	γ	0.1	1.9437	1.7817	1.9292	0.8675	2
	κ	0.1	0.1432	0.1157	0.1630	0.1566	0.2

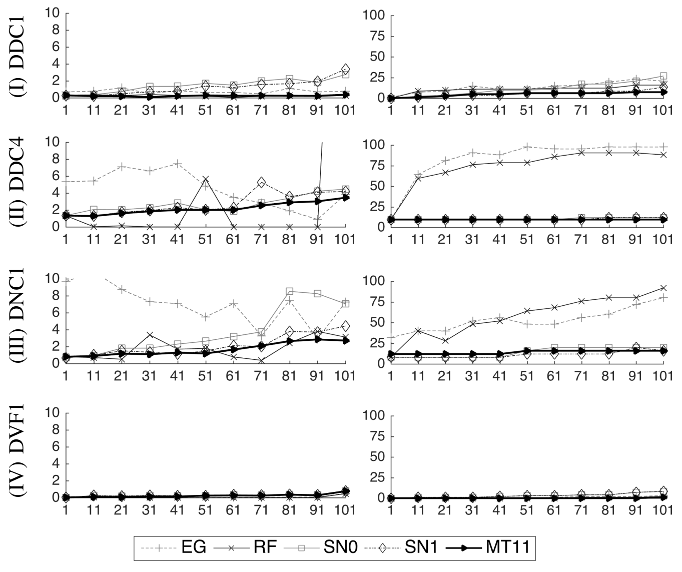
$\ell = 19$, $\alpha = 0.3$, and $\rho = 10$ for all data sets, based on our intuition on the behavior of the algorithm.

For reference algorithms, we systematically determined their optimal parameters *separately for each data set* by finding the parameter settings that minimizes the mean of $\mathcal{E}_\delta$ and $\mathcal{L}_c$ at the last tracked frame. We used the Simulated Annealing (SA) algorithm [36] for this minimization. The parameter search range and initial values used as input into SA were selected based on insights from each algorithm's original paper and on intuition that came with trial-and-error experimentation. Given the long running times needed to track an entire data set (Section III-H), each iteration of SA was performed by tracking a subset of cells over 11 frames. By comparing tracking runs on the entire data set with both optimal and initial parameter settings, we verified that the optimal parameters obtained from this scheme generalized to superior performance on the entire data set. Table I shows the search range and optimal values found using SA for each data set and algorithm. These optimal values were used for all performance evaluations. For EG, `tac` is the threshold on the maximum area change of a tracked cell between frames, `tlb` is the number of previous frames to consider for region matching, and `tcd` is the threshold on the maximum centroid displacement of a tracked cell between frames. For RF, λ_2, λ_1, w, and γ are described in the original paper [8]. For SN0 and SN1, α, β, and γ are described in the original paper [32], and κ scales the gradient of the external energy term E_{ext} for finer control of the optimization procedure.

E. Tracking Results

Figures 3 and 5 show the performance of all algorithms on all data sets. We have also included four supplementary movies in the AVI format, corresponding to the four data sets, that illustrate per-frame tracking results with our algorithm. This will be available at http://ieeexplore.ieee.org.

DDC1 is characterized by strong signal and distinct cell boundaries (row I in Figure 3). EG and RF performed very



(a) δ-region error ($\mathcal{E}_\delta$) (b) % lost cells ($\mathcal{L}_c$)

Fig. 5. Tracking evaluation that quantifies the performance of each algorithm on each data set. Plots show the (a) δ-region error and (b) average % lost cells measures on the y-axes with respect to image frames on the x-axes.

well because watershed segmentation performed very well. Even so, cells gradually got lost over the course of tracking, resulting in $\mathcal{L}_c > 10\%$ at the last frame for both methods (plot Ib in Figure 5). SN0 and SN1 also lost a number of cells ($\mathcal{L}_c > 10\%$) due to the active contours losing track of boundaries and cells shrinking to zero area. Contours in SN0 and SN1 may also drop part of a boundary (arrows in images Ie and If in Figure 3), resulting in $\mathcal{E}_\delta > 2.5$ (plot Ia in Figure 5). In contrast, MT did not experience these problems, resulting in good performance on both measures ($\mathcal{E}_\delta < 0.5$ and $\mathcal{L}_c < 10\%$). MT performed the best on $\mathcal{L}_c$, and only RF performed slightly better than MT on $\mathcal{E}_\delta$ due to accurate watershed segmentation of its remaining, tracked cells.

DDC4 is characterized by low SNR (row II in Figure 3) and large image deformations between adjacent frames. SN0, SN1, and MT all exhibited comparable results on $\mathcal{E}_\delta$, although our method performed consistently better than SN0 or SN1 on this measure after frame 71. All three methods experienced tracking drift due to visible particles that float in the background, but the drift was less for MT (arrows in row II of Figure 3). The weak and variable signal of DDC4 also resulted in very inconsistent segmentations, leading to the frequent additions of false membranes or loss of true membranes between adjacent frames. As a consequence, EG and RF lost track of a significant number of cells. Indeed, EG and RF both lost over 50% of cells by frame 51, while MT did not lose any cells after the first frame (plot IIb in Figure 5 and images IIc, IId, IIg in Figure 3).

DNC1 is characterized by cell boundaries with significant curvature, variable signal, and close proximity to each other (row III in Figure 3). As a consequence, SN0 frequently lost track of the true boundaries (image IIIe of Figure 3), resulting in $\mathcal{E}_\delta > 6$. In contrast, the addition of SIFT flow in SN1 helped to avoid this problem, allowing SN1 to achieve a more competitive $\mathcal{E}_\delta$ score. The high-curvature boundaries also negatively affected the performance of EG on $\mathcal{E}_\delta$, because this method represents each cell as a polygon and thus poorly approximates non-linear boundaries. At the same time, the variable signal resulted in frequent segmentation errors, causing EG and RF to lose a significant number of cells (images IIIc, IIId of Figure 3). By the final frame, both retained fewer than 25% of cells originally in the embryo. MT was robust to these problems that afflicted reference methods and maintained a consistent $\mathcal{E}_\delta < 4$ and $\mathcal{L}_c < 10\%$ (row III of Figure 5).

DVF1 is characterized by wide boundaries, few background artifacts, and small boundary deformations (row IV in Figure 3). Edges are clearly defined and segmentation performed almost perfectly. Thus, all evaluated algorithms performed almost perfectly (row IV in Figure 5). There was a tendency in SN0 and SN1 to disregard sharp corners (images IVe, IVf of Figure 3), but the relatively wide membranes still allowed the contours to accurately trace out each cell for both methods.

One outstanding feature of our method is its robustness. EG and RF have the disadvantage that tracking accuracy completely relies on the quality and consistency of a general-purpose image segmentation algorithm. This resulted in poor performance on $\mathcal{L}_c$, especially on DDC4 and DNC1 ($\mathcal{L}_c > 50\%$ by frame 101 for both data sets). SN0 and SN1 have the disadvantage of curves sometimes detaching from edges by a wide margin. This resulted in notably worse performance on $\mathcal{E}_\delta$ compared to that of MT on DDC1, DDC4, and DNC1. Our algorithm is robust to all these problems, resulting in superior region tracking while accurately retaining almost all cells for the entire duration of tracking. On all four data sets, our method exhibited $\mathcal{E}_\delta < 4$ and $\mathcal{L}_c < 17\%$ (Figure 5).

Our algorithm is also robust to the choice of parameters. Whereas the parameters of reference methods were optimized to each data set, we used the same parameters for all data sets, even though each data set exhibited significantly different SNR, membrane shapes, and intensity profiles. Section III-F demonstrates the robustness of our method's parameters further through a quantitative sensitivity analysis.

F. Parameter Selection and Sensitivity

This section presents an empirical evaluation of the sensitivity of tracking-result quality to parameter values for our method, aiming to develop heuristics for parameter value selection. We started with a set of reference parameter settings and tracked the DDC4 data set several times, each time varying one parameter while keeping all others fixed. We experimented with two different reference parameter settings. Figures 6 and 7 illustrate the results of our investigation.

1) ω and ℓ: First, we suggest that the size of vertex and edge windows, tuned by ω and ℓ, respectively, should not be too small. This is reasonable because smaller windows may be blind to large displacements and are more likely to be biased by noise in their limited supports. On the other hand, windows that are too large may incorrectly include external structures in their supports as would happen, for instance, when nearby edges have overlapping windows (Figure 6).

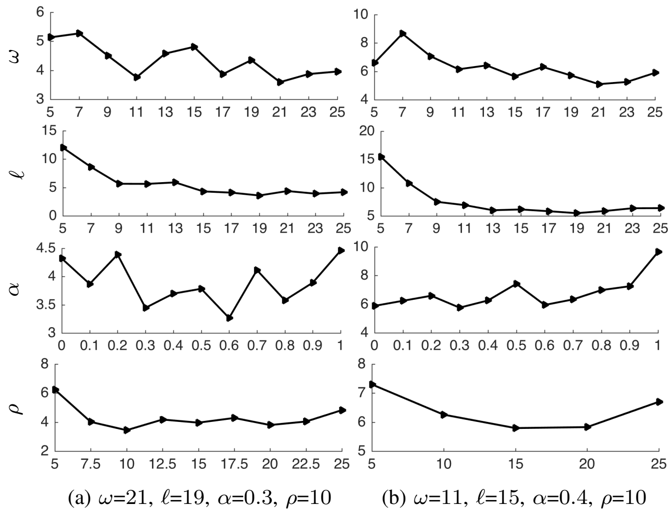


Fig. 6. Sensitivity analysis for DDC4 using the $\mathcal{E}_\delta$ measure, by varying one parameter (rows) while keeping all others fixed. Results from two parameter settings are shown: (a) the parameter setting used for comparison with other algorithms in Section III-E and (b) another setting with good performance but smaller vertex and edge window support. Plots for the $\mathcal{L}_c$ measure are omitted because their changes were negligible with respect to all parameters in this figure. Each data point represents the error at the last tracked frame (101).

2) α: We see that $\alpha < 1$ generally exhibited much better performance, which is reasonable because edge costs provide important information on the graph deformations (Figure 6). We recommend selecting $0.3 < \alpha < 0.7$ for consistently good performance regardless of the values of the other parameters. Nevertheless, we recommend a lower α within this range for images with weaker vertex signal and stronger edge signal, and a larger α within this range when the opposite is true.

3) ρ: The distance ρ between control points should also be neither too large nor too small (Figure 6). This reflects the fact that a ρ that is too small (too many control points) increases the degrees of freedom on the edge shape, which both unnecessarily inflates the size of the tracking optimization problem and allows the edge spline to overfit to the noise neighboring the membrane junctions. On the other hand, a ρ that is too large (too few control points) over-constrains the shape of the edge, preventing the spline from accurately modeling finer features of the membrane junction. However, while $10 < \rho < 20$ is optimal for DDC4, the best value for a particular data set ultimately depends on its image resolution and the boundary shape of its tracked objects.

4) `hminima`: We also investigated the effects of initial segmentation quality on tracking-result quality by evaluating tracking performance with respect to varying the `hminima` parameter introduced in Section III-B. The results in Figure 7 quantify the natural notion that good model initialization leads to good tracking. `hminima` values between 7 and 11 achieved the best results on both measures, corresponding to segmentations that are neither over-segmented nor under-segmented. Furthermore, we note that `hminima` $= 11$, the value selected *a priori* to yield the visibly best results for this data set, corresponded well to a competitive tracking result.

This analysis also provides quantitative evidence for the in-

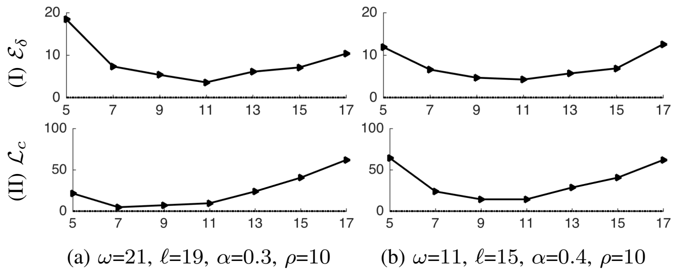


Fig. 7. Sensitivity analysis of tracking performance to segmentation quality on the DDC4 data set, by plotting the effects of varying the hminima parameter on (I) δ-region error and (II) % lost cells. The two columns display results for tracking trials with two different parameter settings (same as those in Figure 6). Each data point represents the error at the last tracked frame (101). The dotted, horizontal line in each plot (all near 0 error) corresponds to the error of tracking with a manually segmented first frame for initialization.

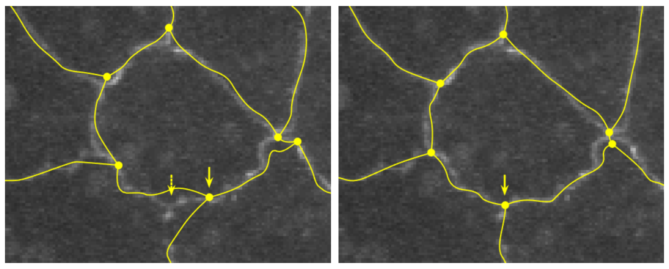


Fig. 8. Improved tracking from edge cost, illustrated on the same cell as in Figure 2, in frame 51. The solid circles overlaid on the images represent tracked vertices, and curves extending from the vertices represent tracked edges. Left: $\alpha = 1$. Note the significant vertex drift. The solid arrow points to a specific tracked vertex location and the dashed arrow points to the true location of that vertex. Right: $\alpha = 0.4$. The vertices are tracked correctly, with the solid arrow pointing to the (same) true and tracked vertex location. (Best viewed when magnified.)

tuition that our algorithm may perform much better if its graph model is initialized from a ground truth segmentation of the first frame, rather than from the potentially imperfect results of an automated segmentation algorithm. Indeed, the dotted, horizontal line in each plot of Figure 7, which correspond to error in tracking with a manually segmented first frame for initialization, is noticeably lower than all other data points in its plot (all of which correspond to error in tracking with watershed-segmented first frames).

G. Edge Contributions

An important novelty of our method is the use of edge information to improve the tracking of vertices. In this section, we provide additional insight into how the addition of edge costs leads to better tracking compared to tracking using vertex costs alone (see Figure 8).

Edges help guide the vertices to their correct positions, especially when vertices have very little signal or the image contains significant deformations between frames that confuse the vertex tracker. Tracking with only vertex cost ($\alpha = 1$) results in vertices that drift along edges, eventually preventing the correct tracking of deforming membranes. Thus, the locally optimal solution computed for each vertex is not the globally optimal solution that minimizes the overall *graph cost* of the system. On the other hand, $\alpha < 1$ (the instance with $\alpha = 0.4$ is displayed in Figure 8) encourages the vertices to converge to an optimal solution according to a more global criterion in which both vertices and edges are correctly outlined.

H. Computation Time

Table II presents the time needed to track each data set, recorded using MATLAB R2015a on Ubuntu 14.04.3 LTS running on twelve Intel Xeon processors clocked between 2.40-2.66 GHz. We found that RF took the most time on DDC1, DNC1, and DVF1. SN1 took the most time on DDC4. The computation time for our method (MT), while less than the maximum for each data set, was in a comparable range. Profiler results for our method indicate the significant number of image interpolations using the MATLAB function interp2 to be our primary bottleneck. As a consequence, our method takes significantly less time to track on down-sampled versions of the data sets.

TABLE II
RUNNING TIME (IN SECONDS) FOR DIFFERENT ALGORITHMS (COLUMNS) ON EACH DATA SET (ROWS).

Data set	EG	RF	SN0	SN1	MT
DDC1	162	38052	1799	8139	18162
DDC4	136	17746	22264	22548	15685
DNC1	33	9380	4610	5457	8951
DVF1	161	61482	1097	2987	14917

I. Limitations of Our Approach

Our algorithm can handle the disappearance of cells (such as through apoptosis), but not cell divisions. However, a wider variety of graph topology changes can be cleanly addressed by improving the UPDATEGRAPHTOPOLOGY function in Algorithm 1. Another limitation is that because the algorithm performs a local search, it may fail upon large, sudden changes in the intensity profile of the image.

IV. CONCLUSION

We develop a robust, optionally fully-automatic method that utilizes least-squares photometric cost functions on a deformable graph model to track epithelial sheets with cells separated by a network of membrane junctions. We demonstrate consistently superior performance compared to four previously published methods on four heterogeneous data sets. This method may be an important tool especially in developmental biology by helping to characterize these commonplace biological structures from the tissue to sub-cellular level.

Future work entails making our method more efficient for quick results on a personal computer and supporting a greater variety of graph topology changes to model biological events such as cell divisions. Our proposed method is not specific to biological membranes, and we are interested in extending our graph tracking methodology to other domains.

Acknowledgments

This material is based upon work supported by the NSF under Grants No. IIS-1208245 and CCF-1513816. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. We thank Daniel Kiehart and other authors of [4] for providing the dorsal closure data sets (collected under his NIH grant GM033830) and offering insightful discussions on automated cell tracking from a biological perspective. We also thank the authors of [5] for providing the dorsal fold data set and the authors of [6] and [8] for providing their source code. We are grateful to the anonymous reviewers for comments and recommendations that greatly improved this article.

References

[1] P. J. Keller, "Imaging morphogenesis: Technological advances and biological insights," *Science*, vol. 340, no. 6137, pp. 1 234 168.1–10, 2013.

[2] D. J. Stephens and V. J. Allan, "Light microscopy techniques for live cell imaging," *Science*, vol. 300, no. 5616, pp. 82–86, 2003.

[3] J. R. Mansfield, K. W. Gossage, C. C. Hoyt, and R. M. Levenson, "Autofluorescence removal, multiplexing, and automated analysis methods for in-vivo fluorescence imaging," *Journal of Biomedical Optics*, vol. 10, no. 4, pp. 041 207.1–9, 2005.

[4] A. R. Wells, R. S. Zou, U. S. Tulu, A. C. Sokolow, J. M. Crawford, G. S. Edwards, and D. P. Kiehart, "Complete canthi removal reveals that forces from the amnioserosa alone are sufficient to drive dorsal closure in Drosophila," *Molecular Biology of the Cell*, vol. 25, no. 22, pp. 3552–3568, 2014.

[5] Z. Khan, Y.-C. Wang, E. F. Wieschaus, and M. Kaschube, "Quantitative 4D analyses of epithelial folding during Drosophila gastrulation," *Development*, vol. 141, no. 14, pp. 2895–2900, 2014.

[6] M. A. Gelbart, B. He, A. C. Martin, S. Y. Thiberge, E. F. Wieschaus, and M. Kaschube, "Volume conservation principle involved in cell lengthening and nucleus movement during tissue morphogenesis," *Proceedings of the National Academy of Sciences*, vol. 109, no. 47, pp. 19 298–19 303, 2012.

[7] S. L. Haigo and D. Bilder, "Global tissue revolutions in a morphogenetic movement controlling elongation," *Science*, vol. 331, no. 6020, pp. 1071–1074, 2011.

[8] A. Chakraborty and A. K. Roy-Chowdhury, "Context aware spatio-temporal cell tracking in densely packed multilayer tissues," *Medical Image Analysis*, vol. 19, no. 1, pp. 149–163, 2015.

[9] C.-Y. Lee, S. Kang, A. D. Chisholm, and P. C. Cosman, "Automated cell junction tracking with modified active contours guided by SIFT flow," in *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*. IEEE, 2014, pp. 290–293.

[10] X. Chen, X. Zhou, and S. T. Wong, "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 4, pp. 762–766, 2006.

[11] M. Liu and P. Xiang, "Automated segmentation and tracking of SAM cells," in *Pattern Recognition*. Springer, 2014, pp. 382–391.

[12] F. Yang, M. A. Mackey, F. Ianzini, G. Gallardo, and M. Sonka, "Cell segmentation, tracking, and mitosis detection using temporal context," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005*. Springer, 2005, pp. 302–309.

[13] R. Bise, Z. Yin, and T. Kanade, "Reliable cell tracking by global data association," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, 2011, pp. 1004–1010.

[14] N. Harder, F. Mora-Bermúdez, W. J. Godinez, J. Ellenberg, R. Eils, and K. Rohr, "Automated analysis of the mitotic phases of human cells in 3D fluorescence microscopy image sequences," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*. Springer, 2006, pp. 840–848.

[15] D. Padfield, J. Rittscher, and B. Roysam, "Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis," *Medical Image Analysis*, vol. 15, no. 4, pp. 650–668, 2011.

[16] X. Yang, H. Li, and X. Zhou, "Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and Kalman filter in time-lapse microscopy," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 11, pp. 2405–2414, 2006.

[17] D. Dormann, T. Libotte, C. J. Weijer, and T. Bretschneider, "Simultaneous quantification of cell motility and protein-membrane-association using active contours," *Cell Motility and the Cytoskeleton*, vol. 52, no. 4, pp. 221–230, 2002.

[18] F. Leymarie and M. D. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 617–634, 1993.

[19] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, "Cell population tracking and lineage construction with spatiotemporal context," *Medical Image Analysis*, vol. 12, no. 5, pp. 546–566, 2008.

[20] N. Ray, S. T. Acton, and K. Ley, "Tracking leukocytes in vivo with shape and size constrained active contours," *IEEE Transactions on Medical Imaging*, vol. 21, no. 10, pp. 1222–1235, 2002.

[21] C. Zimmer, E. Labruyere, V. Meas-Yedid, N. Guillen, and J.-C. Olivo-Marin, "Segmentation and tracking of migrating cells in videomicroscopy with parametric active contours: A tool for cell-based drug testing," *IEEE Transactions on Medical Imaging*, vol. 21, no. 10, pp. 1212–1221, 2002.

[22] A. Dufour, V. Shinin, S. Tajbakhsh, N. Guillén-Aghion, J.-C. Olivo-Marin, and C. Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1396–1410, 2005.

[23] O. Dzyubachyk, W. Van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, "Advanced level-set-based cell tracking in time-lapse fluorescence microscopy," *IEEE Transactions on Medical Imaging*, vol. 29, no. 3, pp. 852–867, 2010.

[24] D. P. Mukherjee, N. Ray, and S. T. Acton, "Level set analysis for leukocyte detection and tracking," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 562–572, 2004.

[25] O. Debeir, P. V. Ham, R. Kiss, and C. Decaestecker, "Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes," *IEEE Transactions on Medical Imaging*, vol. 24, no. 6, pp. 697–711, 2005.

[26] A. Sokolow, Y. Toyama, D. P. Kiehart, and G. S. Edwards, "Cell ingression and apical shape oscillations during dorsal closure in Drosophila," *Biophysical Journal*, vol. 102, no. 5, pp. 969–979, 2012.

[27] C. Liu, J. Yuen, and A. Torralba, "SIFT flow: Dense correspondence across scenes and its applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 978–994, 2011.

[28] A. Blake and M. Isard, *Active Contours*. New York, NY: Springer, 1998.

[29] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision." in *International Joint Conference on Artificial Intelligence*, vol. 81, 1981, pp. 674–679.

[30] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY: Springer, 1999.

[31] G. L. Hunter, J. M. Crawford, J. Z. Genkins, and D. P. Kiehart, "Ion channels contribute to the regulation of cell sheet forces during Drosophila dorsal closure," *Development*, vol. 141, no. 2, pp. 325–334, 2014.

[32] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[33] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Prentice Hall, 2007.

[34] S. Pop, A. Dufour, and J.-C. Olivo-Marin, "Image filtering using anisotropic structure tensor for cell membrane enhancement in 3D microscopy," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 2041–2044.

[35] V. Prasath, R. Pelapur, O. Glinskii, V. Glinsky, V. Huxley, and K. Palaniappan, "Multiscale tensor anisotropic filtering of fluorescence microscopy for denoising microvasculature," in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, April 2015, pp. 540–543.

[36] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.