Tracking *Tetrahymena Pyriformis* Cells using Decision Trees

Quan Wang[1], Yan Ou[1], A. Agung Julius[1], Kim L. Boyer[1], Min Jun Kim[2]
[1]Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2]Drexel University, Philadelphia, PA 19104, USA

Abstract

Matching cells over time has long been the most difficult step in cell tracking. In this paper, we approach this problem by recasting it as a classification problem. We construct a feature set for each cell, and compute a feature difference vector between a cell in the current frame and a cell in a previous frame. Then we determine whether the two cells represent the same cell over time by training decision trees as our binary classifiers. With the output of decision trees, we are able to formulate an assignment problem for our cell association task and solve it using a modified version of the Hungarian algorithm.

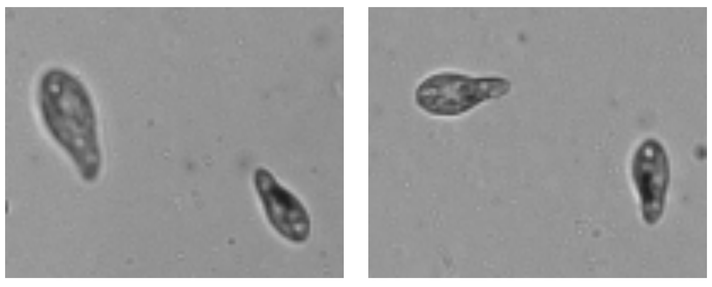
Basic Idea

We maintain a list of all the cells that have appeared in previous frames, and we perform a background subtraction on the current frame to segment current cell regions. For a pair of one cell region in the current frame and one cell in the list, we classify this pair to two classes: "same cell" and "different cells". The classifier we use is decision tree, which is trained on manually labeled videos.

In testing, with the probabilities of each cell pair given by the decision trees, we are able to construct an association matrix. By applying the modified Hungarian algorithm to solve the optimization problem, each cell region in the current frame is determined as: (1) a cell that has appeared before; (2) a new cell entering the frame; or (3) a region of multiple cells overlapping each other.

Data Description

The videos were captured at 8 frames per second, where each frame is an 8-bit gray-level image of size 640×512. The typical size of a *T. pyriformis* cell is between 150 and 400 pixels, and the typical speed is between 5 and 40 pixels per frame. The pixel spacing is 2.32 μm. See examples of the *T. pyriformis* cell appearance below.



Feature Extraction

Spatial features include a cell's area, centroid, and *n*th order normalized inertia:

$$J_n(C) = \frac{\sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{x}_c||^n}{N^{1+n/2}}$$

Histogram features are the statistics of the pixel intensities, including the mean, the standard deviation, the skewness, the kurtosis, and the *n*th order root of the *n*th order central moment:

$$M_n(C) = \sqrt[n]{E\left[(I(\mathbf{x}_i) - \mu(C))^n\right]}$$

Composite features use both spatial information and pixel intensities at the same time to capture the spatial distribution of pixel intensities, including polynomial distribution feature:

$$P_n(C) = \frac{1}{N^{1+n/2}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{x}_c||^n \cdot I(\mathbf{x}_i)$$

and Gaussian distribution feature:

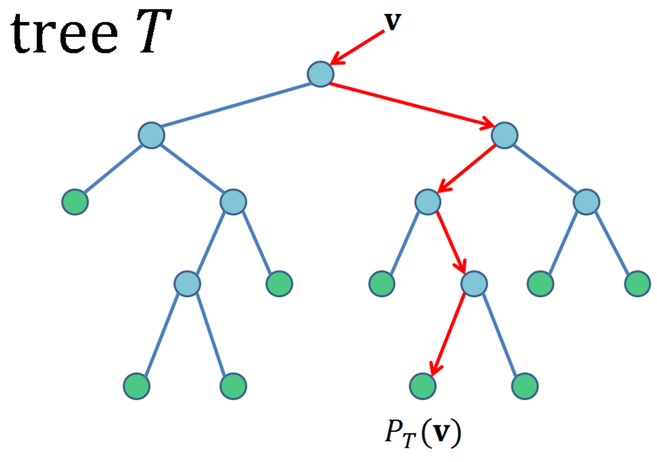
$$G_n(C) = \frac{1}{N} \sum_{i=1}^{N} \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_c||^2}{2n^2}\right) \cdot I(\mathbf{x}_i)$$

We can choose different values of *n* to generate sufficient features.

Decision Trees

The input to the decision tree is a 21-dimensional feature difference vector. Each entry of this vector is the absolute value of the difference of some feature of two cells. The decision trees are trained with the maximal entropy reduction procedure. We train two trees: T_1 and T_2. T_2 is trained using truncated feature difference vectors without center position information.

Depth of Tree	Error Rate of T_1	Error Rate of T_2
5	1.48%	14.02%
6	1.46%	13.91%
7	1.69%	12.95%
8	1.37%	12.49%
9	1.54%	13.07%
10	1.55%	13.61%



Cell Association

With N_1 cells in the current frame and N_2 cells in the list, we define an $N_1 \times (N_2+2)$ association matrix A. Each entry of A is a confidence of associating a current cell to an old cell, a new cell, or an overlapping region. Since more than one cell can be considered as new cell or overlapping region, to maximize the overall association confidence, we have this assignment problem:

For an association list $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_{N1})$, find

$$\boldsymbol{\zeta}^* = \underset{\boldsymbol{\zeta}}{\operatorname{argmax}} \sum_{i=1}^{N_1} A_{i,\zeta_i}$$

such that for any $1 \le j \le N_2$, there is at most one i that satisfies $\zeta_i = j$.

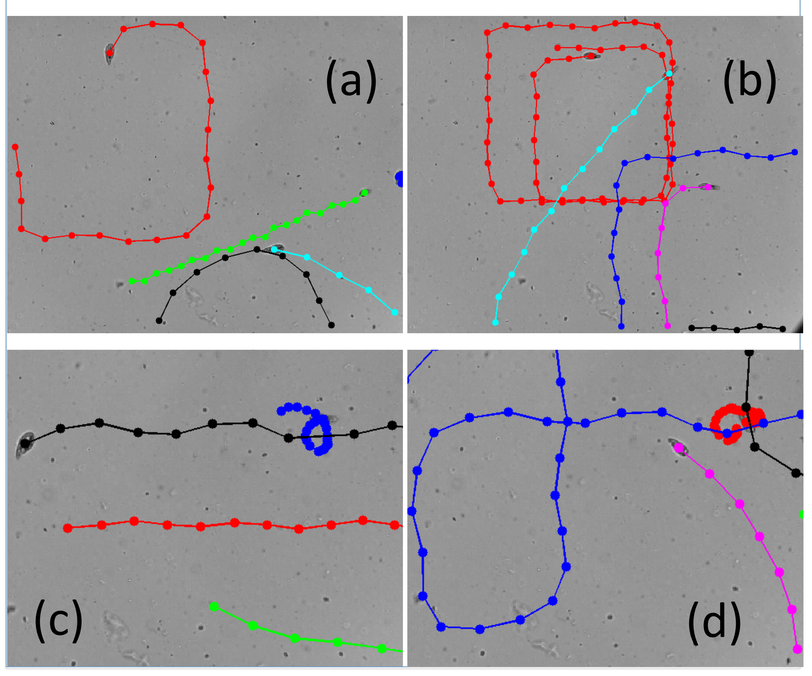
Modified Hungarian Alg.

To solve the above mentioned derivative assignment problem, we propose a modified Hungarian algorithm to get an approximate solution in linear time. The performance on randomly generated matrices is given below (each experiment is repeated 20 times).

N_1	N_2	Brute force time	Modified Hungarian time	Modified Hungarian error
3	3	8.018 ms	0.777 ms	0.0086%
4	4	59.43 ms	0.969 ms	0.2651%
5	5	523.0 ms	1.319 ms	0.3206%
6	6	5.164 s	1.532 ms	0.5257%
7	7	58.78 s	1.883 ms	0.5032%
8	8	746.3 s	2.304 ms	0.2569%

Resulting Trajectories

(a) Cells getting very close. (b) Many cells in the frame. (c) and (d) Cells occluding each other.



Electrical, Computer, and Systems Engineering at Rensselaer