Tracking *Tetrahymena Pyriformis* Cells using Decision Trees

Quan Wang, Yan Ou, A. Agung Julius, Kim L. Boyer
Rensselaer Polytechnic Institute
wangq10@rpi.edu

Min Jun Kim
Drexel University
mkim@coe.drexel.edu

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.

1 Introduction

Researchers have been creating artificial magnetotactic *Tetrahymena pyriformis* (*T. pyriformis*) cells by the internalization of iron oxide nano particles, and controlling them with a time-varying external magnetic field [2, 5]. To perform the multi-cell control task, it is necessary to track the real-time position of each cell and use it as the feedback for the control system. However, this problem is very difficult because: the *T. pyriformis* cells are moving fast in the field of view; several cells may overlap and occlude each other; the appearances of different cells can be similar; and the same cell may change in appearance over time [4].

Decision tree was proposed as a powerful machine learning technique, which can be used for both classification and regression, and has been successfully applied on practical systems such as the Kinect gaming platform [6]. Since the decision tree can take very complicated features as its input and at the same time, once a decision tree is trained, the decision procedure can be very fast, it is ideal for real-time classification. We use decision trees as a classifier to determine whether regions segmented in neighboring and non-neighboring frames represent the same *T. pyriformis* cell.

2 Background Subtraction

To extract the foreground regions of cells, we first perform a background subtraction at each frame. There are a number of well-known background subtraction algorithms such as adaptive Gaussian mixture models. Since the background in our video is simple and consistent over time, we simply apply a median filter on the time dimension for the first 20 frames to obtain the background image. At each frame, we subtract the background image from the current frame and take the absolute value to obtain a difference image. Then we apply thresholding, fill the morphological holes, and extract the connected regions as candidate *T. pyriformis* cells.

3 Feature Extraction

After the region of each cell is segmented, we extract features for each cell according to their shapes, gray-level intensities, and the combination of both. Examples of the *T. pyriformis* cell appearance in our video are shown in Figure 1 (pixel spacing $= 2.32/2.32\,\mu$m). Assume a cell C in one frame consists of N pixels $I(\mathbf{x}_i)$, where $i = 1, 2, \ldots, N$, and $\mathbf{x}_i = (x_i, y_i)$ are the coordinates of the ith pixel of C. We can define some useful features for this cell, as follows.

Spatial Features The cell's area N and centroid $\mathbf{x}_c = (x_c, y_c)$, where $x_c = \overline{x_i}$ and $y_c = \overline{y_i}$, are the most basic features. Another useful spatial feature is the nth order normalized inertia [1], which measures the circularity of the shape, and is defined as

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

Histogram Features Using only pixel intensities without spatial positions, we can compute several his-

togram features. First we have the mean and the standard deviation:

$$\mu(C) \;=\; \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{x}_i), \tag{2}$$

$$\sigma(C) \;=\; \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(I(\mathbf{x}_i)-\mu(C)\right)^2}. \tag{3}$$

Then we have the standard moments such as the skewness $\gamma(C)$ and the kurtosis $\beta(C)$:

$$\gamma(C) \;=\; E\left[\left(\frac{I(\mathbf{x}_i)-\mu(C)}{\sigma(C)}\right)^3\right], \tag{4}$$

$$\beta(C) \;=\; E\left[\left(\frac{I(\mathbf{x}_i)-\mu(C)}{\sigma(C)}\right)^4\right], \tag{5}$$

where $E[\cdot]$ denotes expectation. We also use the nth order root of the nth order central moment $M_n(C)$:

$$M_n(C) = \sqrt[n]{E\left[\left(I(\mathbf{x}_i)-\mu(C)\right)^n\right]}. \tag{6}$$

Composite Features Finally we can compute some features that use both spatial information and pixel intensities at the same time to capture the spatial distribution of pixel intensities. We first define the polynomial distribution feature $P_n(C)$:

$$P_n(C) = \frac{1}{N^{1+n/2}}\sum_{i=1}^{N}\|\mathbf{x}_i-\mathbf{x}_c\|^n \cdot I(\mathbf{x}_i). \tag{7}$$

Then we define the Gaussian distribution feature $G_n(C)$:

$$G_n(C) = \frac{1}{N}\sum_{i=1}^{N}\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_c\|^2}{2n^2}\right)\cdot I(\mathbf{x}_i). \tag{8}$$

These composite features can be viewed as weighted means of pixel intensities. For example, $P_n(C)$ gives larger weights to pixels far away from the center while $G_n(C)$ gives larger weights to pixels near the center.

4 Decision Trees

To use decision trees for the cell association problem, we first need to map the cell association problem to a classification problem. For each cell L in a previous frame and each cell C in the current frame, we need to know whether they are the same cell. Thus the classification problem is to develop a binary classifier f such that $f(C,L)=1$ if C and L are the same cell, and $f(C,L)=0$ if they are not.
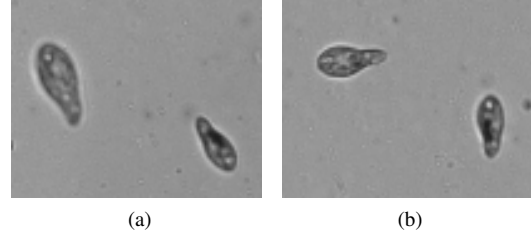

(a)　　　　(b)

Figure 1: Examples of the *T. pyriformis* cell appearance.

4.1 Feature Difference Vector

To simplify the input to the classifier, we define a 23-dimensional feature difference vector $\mathbf{v}(C,L) = (v_1, v_2, \ldots, v_{23})$ for C and L. Each entry of this feature difference vector reflects the difference of the two cells in some aspect. v_1 is the distance between the center of C and the predicted center of L in the current frame. A simple constant velocity assumption is used for the prediction. For example, suppose the current frame is the kth frame and the center of C is $\mathbf{x}_c(C)$, the center of L at the $(k-1)$th frame was $\mathbf{x}_c(L)$ and at the $(k-2)$th frame was $\mathbf{x}'_c(L)$. Then the predicted center of L at the current frame is $\mathbf{x}_p(L) = 2\mathbf{x}_c(L) - \mathbf{x}'_c(L)$, and v_1 is defined as

$$v_1 = \|\mathbf{x}_c(C) - \mathbf{x}_p(L)\|. \tag{9}$$

v_2 is defined as the distance between the centers of the two cells:

$$v_2 = \|\mathbf{x}_c(C) - \mathbf{x}_c(L)\|. \tag{10}$$

Each of the other 21 entries of $\mathbf{v}(C,L)$ is defined as the absolute value of the difference of some feature of C and L. For example, let $N(C)$ and $N(L)$ be the areas of C and L, respectively. Then we have

$$v_3 \;=\; |N(C) - N(L)|, \tag{11}$$
$$v_4 \;=\; |\mu(C) - \mu(L)|. \tag{12}$$

Similarly, v_5 to v_7 are defined using Equations (3) to (5) respectively; v_8 to v_{11} are defined using (6) for $n = 3, 4, 5, 6$; v_{12} to v_{15} are defined using (1) for $n = 1, 2, 3, 0.5$; v_{16} to v_{19} are defined using (7) for $n = 1, 2, 3, 0.5$; and v_{20} to v_{23} are defined using (8) for $n = 2, 4, 6, 8$. With these definitions, our classifier f can be viewed as a mapping from the 23-dimensional space to the $\{0, 1\}$ set. Thus we denote it as $f(\mathbf{v}(C,L))$.

4.2 Training a Decision Tree

As shown in Figure 2, a decision tree T consists of leaf nodes and split (non-leaf) nodes. Each split node consists of a threshold τ and an indicator k for a feature v_k, where $k = 1, 2, \ldots, 23$. To classify the feature
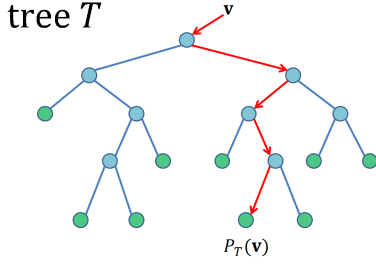
tree T

$P_T(\mathbf{v})$

Figure 2: Decision tree example. A decision tree T consists of split nodes (blue) and leaf nodes (green). The red arrows show the branching path for an input $\mathbf{v}$.

difference vector $\mathbf{v}$, one starts from the root and repeatedly compares its kth entry v_k with the threshold τ to decide whether to branch to the left sub-tree (if $v_k \leq \tau$) or right sub-tree (if $v_k > \tau$). The leaf node is a quantity $P_T(\mathbf{v})$ indicating the probability of $f(\mathbf{v}(C, L)) = 1$.

Since it is easy for a human to distinguish whether two regions in different frames represent the same cell, we can manually assign labels $\{y\}$ to cell pairs from neighboring frames in training videos and associate these labels with extracted feature difference vectors $\{\mathbf{v}\}$. In this way we build our training data set $D = \{(\mathbf{v}, y)\}$, where $y = f(\mathbf{v})$.

The training process follows a maximal entropy reduction procedure [6]:

1. Beginning from the root, consider a large set of splitting candidates (k, τ) which covers all possible k and provides sufficiently delicate subdivisions for each v_k.

2. For each candidate (k, τ), we partition the training set $D = \{(\mathbf{v}, y)\}$ into two subsets:

$$D_l(k, \tau) = \{(\mathbf{v}, y)|v_k \leq \tau\}, \qquad (13)$$
$$D_r(k, \tau) = D \setminus D_l(k, \tau). \qquad (14)$$

3. Find the candidate (k, τ) that maximizes the entropy reduction $G(k, \tau)$:

$$(k^*, \tau^*) = \underset{(k,\tau)}{\operatorname{argmax}} \, G(k, \tau), \qquad (15)$$

$$G(k, \tau) = H(D) - \frac{|D_l(k, \tau)|}{|D|} H(D_l(k, \tau))$$
$$- \frac{|D_r(k, \tau)|}{|D|} H(D_r(k, \tau)), \quad (16)$$

where $H(\cdot)$ denotes the Shannon entropy and $|\cdot|$ denotes the cardinality of a set.

4. Use (k^*, τ^*) as the feature indicator and threshold at the current split node, and repeat the above steps for the left sub-tree with $D_l(k^*, \tau^*)$ and right sub-tree with $D_r(k^*, \tau^*)$.

5. If the depth reaches the maximum, or the size (or entropy) of the partitioned subset $\widetilde{D}$ at the current node is sufficiently small, then we set this node as a leaf node, and the probability at this node is

$$P_T = \frac{|\{(\mathbf{v}, y) \in \widetilde{D}|y = 1\}|}{|\widetilde{D}|}. \qquad (17)$$

In our work, a tree T_1 is trained using $D = \{(\mathbf{v}, y)\}$, and another tree T_2 is trained using $D' = \{(\mathbf{v}', y)\}$, where $\mathbf{v}' = (v_3, v_4, \dots, v_{23})$ is the truncated feature difference vector. Since T_2 is trained without center position information, we use T_1 for cell association in neighboring frames and T_2 for cell association in non-neighboring frames.

5 Association Rules

To track cells over time, we maintain a list $\{L_j\}$ of cells that have appeared in previous frames. The cell association task is to determine which cell L_j should be associated with the cell C_i in the current frame. Each cell in the list has a status, being *active*, *occluded*, *out*, or *new*.

5.1 Association Matrix

At frame k, if we have N_1 cells $\{C_i\}$ in the current frame and N_2 cells $\{L_j\}$ in the list, we define an $N_1 \times (N_2 + 2)$ association matrix A. Each entry of A represents the confidence of associating C_i with a cell in the list or considering it as a new cell.

If $1 \leq j \leq N_2$ and the status of L_j is *active*, *new* or *occluded*, we define

$$A_{i,j} = P_{T_1}(\mathbf{v}(C_i, L_j)). \qquad (18)$$

Let d_0 denote the maximum possible speed of a *T. pyriformis* cell in pixels per frame, and $d_b(C_i)$ denote the distance from C_i to the image border. A new cell or a re-appearing cell must (re-)enter the image from the image border. Thus for $1 \leq j \leq N_2$, if the status of L_j is *out* and $d_b(C_i) > d_0$, we set $A_{i,j} = -1$; if the status of L_j is *out* and $d_b(C_i) \leq d_0$, we define

$$A_{i,j} = \alpha_1 P_{T_2}(\mathbf{v}'(C_i, L_j)), \qquad (19)$$

where $0 < \alpha_1 < 1$ is a constant denoting the diffidence (as shown in Table 1) of associating cells in non-neighboring frames.

A_{i,N_2+1} is the confidence of considering C_i as a new cell that has not appeared in previous frames. If $d_b(C_i) > d_0$, we set $A_{i,N_2+1} = -1$; if $d_b(C_i) \leq d_0$, we define

$$A_{i,N_2+1} = \alpha_2 \exp(-\lambda_1 d_b^2(C_i)), \qquad (20)$$

where $0 < \alpha_2 < 1$ and $\lambda_1 > 0$ are two constants.

To also include the case when one cell is occluded by another, we use A_{i,N_2+2} to represent the confidence that C_i is a region of more than one cell overlapping. Let $\{L'_j\}$ be a subset of the list containing only cells that have appeared in the $(k-1)$th frame. We define

$$A_{i,N_2+2} = \alpha_3 \sum_{L'_j} e^{-\lambda_2 d_p^2(C_i,L'_j)} - \alpha_3 \sum_{C_j} e^{-\lambda_2 d_c^2(C_i,C_j)},$$
$$(21)$$

where $0 < \alpha_3 < 1$ and $\lambda_2 > 0$ are two constants, $d_c(C_i, C_j)$ is the center distance between C_i and C_j, and $d_p(C_i, L'_j)$ is the distance from the center of C_i to the predicted center of L'_j, which is similar to (9). The right-hand side of (21) can be thought of as the predicted number of cells minus the observed number of cells near C_i.

5.2 Association List

Once the association matrix A is created, we need to associate each C_i (or each row of A) with a column of A. This problem is similar to the standard assignment problem [3], but note that: first, A is generally not a square matrix; second, more than one C_i can be considered as new or being a region of occluded cells. We define an association list $\boldsymbol{\zeta} = (\zeta_1, \zeta_2, \ldots, \zeta_{N_1})$ such that $\zeta_i = j$ if C_i is associated with the jth column of A. Then our problem is to find

$$\boldsymbol{\zeta}^* = \underset{\boldsymbol{\zeta}}{\operatorname{argmax}} \sum_{i=1}^{N_1} A_{i,\zeta_i} \qquad (22)$$

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

5.3 The Modified Hungarian Algorithm

When N_1 and N_2 are both small, the derivative assignment problem (22) can be solved using a brute-force search. But when N_1 and N_2 are large, the computational complexity of brute-force search is at least

$$O\left(\frac{(\max\{N_1, N_2\})!}{|N_1 - N_2|!}\right), \qquad (23)$$

which is unacceptable for real-time tracking. We propose a modified version of the Hungarian algorithm [3]

to obtain a suboptimal solution of (22). The computational complexity of the original Hungarian algorithm is $O((\max\{N_1, N_2\})^3)$, and our modified Hungarian algorithm can achieve a computational complexity of at most $O((\max\{N_1, N_2\})^3 \cdot \min\{N_1, N_2\})$. The modified Hungarian algorithm is given below:

1. For any $1 \leq i \leq N_1$, if the ith row of A satisfies

$$\underset{j}{\operatorname{argmax}} A_{i,j} = j^* > N_2, \qquad (24)$$

   then we associate the ith row with the j^*th column, and delete the ith row from A to obtain a $N'_1 \times (N_2 + 2)$ submatrix A'.

2. Perform the standard Hungarian algorithm on the first N_2 columns of A' to get an association list $\boldsymbol{\zeta}$.

3. For $1 \leq i \leq N'_1$, compute this value:

$$\Omega(A', i) = \max\{A'_{i,N_2+1}, A'_{i,N_2+2}\} - A'_{i,\zeta_i}. \qquad (25)$$

   If $\max_i \Omega(A', i) > 0$, let

$$i^* = \underset{i}{\operatorname{argmax}} \, \Omega(A', i). \qquad (26)$$

   Then we associate the i^*th row of A' with the $(N_2 + 1)$th column if $A'_{i^*,N_2+1} > A'_{i^*,N_2+2}$ or the $(N_2 + 2)$th column if otherwise. Now we delete the i^*th row from A', update N'_1, and repeat steps 2 to 3.

4. If $\max_i \Omega(A', i) \leq 0$, then the current $\boldsymbol{\zeta}$ is the final association result for the current A'.

5.4 Updating the List of Cells

We update the list of cells $\{L_j\}$ according to the association result of each C_i:

- If C_i is associated with a cell L_j in the list ($1 \leq \zeta_i \leq N_2$), we update the status of L_j to *active* and replace all its features with the features of C_i.

- If C_i is considered to be a new cell ($\zeta_i = N_2 + 1$), we simply add it to the list and set its status to *new*.

- If C_i is considered to be a region of overlapping cells ($\zeta_i = N_2 + 2$), we search its neighbourhood within the radius d_0 for unassociated L_j's that have appeared in the $(k-1)$th frame. Then we set the status of these L_j's to *occluded*, and update only their center positions to the center position of C_i.

- For any L_j that appeared in the $(k-1)$th frame and is still unassociated after all the above steps: if it is within the distance d_0 from the image border, we set its status to *out*; otherwise, we associate it with the closest unassociated C_i.
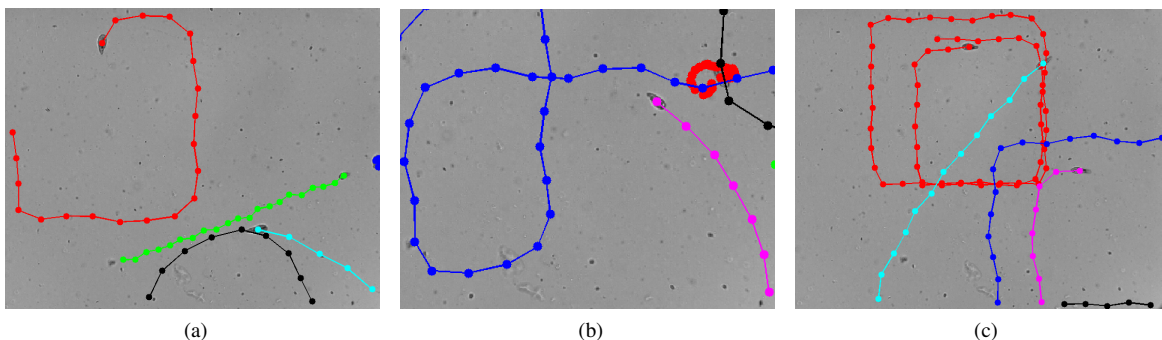
Figure 3: Resulting cell trajectories of our method on testing videos. (a) Cells getting very close. (b) Two cells occluding each other (red *vs.* blue, and red *vs.* black). (c) Many cells appearing in the frame at the same time.

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%

Table 1: Performance of decision trees.

6 Experiments

To build the training set, we manually label the cell regions segmented in 1000 frames from 2 videos, where each frame is an 8-bit gray-level image of size 640×512, and the videos were captured at 8 frames per second [2]. 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. To evaluate the performance of the decision trees T_1 and T_2, we repeat 20 independent experiments for different depth of the trees, and in each experiment we use randomly selected 70% of the cell pairs as the training data, and the rest 30% as the test data. The number of subdivisions of each feature is set to 1000, and the stop-splitting size of $\widetilde{D}$ is set to 20. The classifier f is configured such that $f(\mathbf{v}) = 1$ if $P_T(\mathbf{v}) > 0.5$. The resulting average rates of misclassification are shown in Table 1, and all 23 entries of the feature difference vector $\mathbf{v}$ turn out to be useful.

In our tracking experiments, we use decision trees of depth 8. The parameters α_1, α_2, α_3, λ_1, λ_2 and d_0 are selected empirically by studying the training videos, and the tracking results are not sensitive to minor changes of these parameters. Example cell trajectories obtained by our method are shown in Figure 3. In these examples we set $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, $\alpha_3 = 0.8$, $\lambda_1 = 0.00008$, $\lambda_2 = 0.00005$ and $d_0 = 70$.

7 Discussion and Future Work

This paper has shown a novel cell tracking method using decision trees as classifiers for feature difference vectors. The misclassification rates of our decision trees are very low, and the tracking results of our method are robust against cell occlusions. Future work includes evaluating this method on more complicated videos in which more cells exist in the frame and occlusions of many cells can happen. Finally we will integrate our algorithm into the real-time *T. pyriformis* control system.

References

[1] A. Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, 25(4):373–380, 1979.

[2] D. H. Kim, U. K. Cheang, L. Kohidai, D. Byun, and M. J. Kim. Artificial magnetotactic motion control of *Tetrahymena pyriformis* using ferromagnetic nanoparticles: A tool for fabrication of microbiorobots. *Applied Physics Letters*, 97(17):173702, 2010.

[3] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[4] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen. Tracking in cell and developmental biology. *Seminars in Cell & Developmental Biology*, 20(8):894–902, 2009.

[5] Y. Ou, D. H. Kim, P. Kim, M. J. Kim, and A. A. Julius. Motion Control of *Tetrahymena Pyriformis* Cells with Artificial Magnetotaxis: Model Predictive Control (MPC) Approach. In *IEEE International Conference on Robotics and Automation*, St. Paul, USA, May 2012.

[6] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*, June 2011.