EFFICIENT ACQUISITION AND LEARNING OF FLUORESCENCE MICROSCOPE DATA MODELS

Charles Jackson[1], *Robert F. Murphy*[1,3] *and Jelena Kovačević*[1,2]

[1] Dept. of Biomedical Eng. and Center for Bioimage Informatics,
[2] Dept. of Electrical and Computer Eng.,
[3] Dept. of Biological Sciences and Dept. of Machine Learning,
Carnegie Mellon University, Pittsburgh, PA, USA

ABSTRACT

We present a method for efficient acquisition of fluorescence microscope datasets, to allow for higher spatial and temporal resolution, and with less damage from photobleaching. Our proposal is to restrict acquisition to regions where we expect to find an object. Given that the objects are continuously moving, we must have an accurate model to describe objects' motion to predict their future locations. We outline a system for learning and applying this motion model, provide details from some simple simulations, and summarize results from more complex applications.

Index Terms— Fluorescence, microscopy, tracking, state space methods, Monte Carlo methods

1. INTRODUCTION

Fluorescence microscopy is one of the most popular tools for live-cell imaging. As the trend in biology tends more and more towards automated systems for high-throughput applications, the amount of image data acquired with this technique is growing rapidly. To observe a cellular process over a sustained period, we take a time series of images, where each image is known as a frame. This paper describes an efficient way to acquire such images, in which we obtain the required information without acquiring the entire field of view.

The first motivation for this work is to enhance resolution. In laser scanning confocal microscopy, images are acquired line-by-line, pixel-by-pixel [1]. We can achieve significant time savings by only imaging those regions where we expect to find an object. These time savings could be then used to increase the frame rate, or to acquire the selected regions at a higher spatial resolution.

The second motivation is to reduce photobleaching and phototoxicity. In fluorescence microscopy, images are acquired by shining excitation light on the specimen to activate fluorescence. However, this can damage the fluorescent signal (photobleaching) [2], as well as the cell itself (phototoxicity) [3], thus limiting the duration over which we can view a cellular process. By reducing the total area acquired in each frame, we reduce the overall exposure to excitation light, hence reducing both photobleaching and phototoxicity.

Although algorithms for efficient image compression or image enhancement are well studied, efficient acquisition of these images is not. In [4], the authors designed an algorithm to reduce the number of pixels sampled in a 2D or 3D image when using a laser scanning confocal microscope. They observed that a large portion of scanning time is spent on low fluorescence regions, which presumably contain little useful information. The approach is then to begin by scanning the field at a low resolution. Each scanned value is examined, and if found to be significant, the area around it is scanned at a higher resolution. The process is repeated iteratively.

Here, we are studying a large number of tiny moving objects over a sustained period. To assist with efficient acquisition, we develop and continually refine a model to describe the objects' motion. In [5], the authors provide algorithms for modeling objects' motions for the purposes of tracking, and although not used directly, their work helped inspire our approach.

Section 2 outlines the framework used for developing the motion models, while the acquisition algorithm is described in Section 3. Section 4 presents a case study of a simple system, while Section 5 outlines some characteristics of more complex models.

2. TRACKING FRAMEWORK

A fundamental part of this work is to learn the motion model for each object, and thus, here, we describe the framework in which we do this. We have taken the same approach as used in [5] for object tracking algorithms.

2.1. State Space Model

The state space model assumes that all the necessary information about a system can be summarized by a set of state variables. For a simple case of object tracking, these state variables could be, for example, the 3D coordinates of the centroids of every object. A more advanced model could also include the size, shape, type of every object, etc.

This system is governed by two fundamental state space equations [6]. Equation (1) describes the present state in terms of the previous state, and (2) describes the observed variables in terms of the state variables:

$$x(t+1) = F(t)x(t) + \nu(t), \quad (1)$$
$$z(t) = H(t)x(t) + \mu(t). \quad (2)$$

Equation (1) shows the state at time t, $x(t)$, evolving to the state at time $t+1$, $x(t+1)$, as governed by the state transition matrix $F(t)$. The equation implicitly assumes a linear system, although a more general form can be used for nonlinear systems. The state noise $\nu(t)$ reflects that the model will not perfectly predict the state transitions.

This work was supported in part by NSF through grant EF-0331657, as well as the PA State Tobacco Settlement, Kamlet-Smith Bioinformatics Grant.

Equation (2) maps the state variables $x(t)$ to the measurement variables $z(t)$. $H(t)$ is the observation matrix (if $H(t)$ were a diagonal matrix then the system would be fully observable), and $\mu(t)$ is the measurement noise. Here we assume perfect measurement.

2.2. Application to Motion Modeling

Our method of modeling the objects' motion follows that of [6], as described in this section. Learning the motion model of an object is equivalent to learning $F(t)$ and the properties of $\nu(t)$ for the object. Because we assume stationary object dynamics, $F(t)$ can be represented simply as F, and the state noise has a constant covariance Q. Every object in the specimen potentially moves under a different motion model. Therefore, we have state transition matrices $F_1, ..., F_m$ and covariances $Q_1, ..., Q_m$ for the m objects of interest. Learning the motion models equates to learning these matrices.

Two restrictions are imposed on the model. The first is that covariance matrices are diagonal (making each dimension independent). The second is that F is restricted to one of the three motion models from [5] to cover most motions observed in practice. These three models are a random walk, F_{RW}, a first-order linear extrapolation, F_{FLE}, and a second-order linear extrapolation, F_{SLE}, and require knowledge of the object's position in up to three successive frames. Therefore, in 2D, we define the state vector as:

$$x(t) = \left[\begin{array}{cccccc} x_t & y_t & x_{t-1} & y_{t-1} & x_{t-2} & y_{t-2} \end{array} \right]^\top,$$

where (x_t, y_t) represent the 2D coordinates of the object at time t. If we were using the FLE model, for example, the state update equation would be:

$$x_{t+1} = 2x_t - x_{t-1} + N(0, \sigma_x), \quad (3)$$
$$y_{t+1} = 2y_t - y_{t-1} + N(0, \sigma_y). \quad (4)$$

3. ALGORITHM OUTLINE

Problem statement. Our goal here is to learn the motion model for each object. The input to the system is the time series of images (frames), and the set of possible motion models that could describe the objects. For each of these motion models, and for each object, the system outputs the relative likelihood that a given motion model describes a given object. As our knowledge of each object's motion becomes more refined, the efficiency of acquisition improves.

If we are aiming for high temporal resolution, then we are trying to reduce the number of pixels that we acquire. This is the goal for the case study presented in this paper. Alternatively, to reduce photobleaching, we would try to reduce the number of times we acquire the object itself. This can be done by reducing the frame rate, and by learning the motion models from a representative subset of the objects instead of acquiring every object. These concepts will be expanded upon in future publications.

Assumptions. We assume that objects are perfectly detected provided that the appropriate region is acquired. This is in contrast to [6], which considers the possibility that an object goes undetected due to background noise, or, alternatively, that background clutter is falsely detected as an object. These considerations will be taken into account in future work.

A second assumption is that each object of interest occupies a single pixel, thus avoiding size and shape considerations. Future algorithms will model the possibly deformable sizes and shapes of objects along with their motion.

Finally, although our system does allow for nonstationary object dynamics, all experiments so far assume that these dynamics are stationary. That is, we assume that the motion models are not changing over time.

Algorithm. The system does not make hard decisions about which motion model an object is operating under, but instead associates a probability with each possible model. We begin by assigning initial probabilites to each model. For example, we could assume that the three models F_{RW}, F_{FLE}, F_{SLE}, are all equally likely. The covariance of the state noise is also a model parameter. We could initially assume that the possible values of the variance (either in the x-direction or the y-direction) are all equally likely, with some upper bound (such as the diameter of the cell). The algorithm then refines these probabilities as it cycles through the following steps (a pseudo code is given in Algorithm 1):

1. Predict distribution of objects: For each possible motion model, we calculate the likelihood of finding each object in any given pixel in the subsequent frame.
2. Acquire pixels according to some policy: For example, to increase temporal resolution, we may choose to acquire a fixed number of pixels that gives the highest probability of capturing each object.
3. Having acquired these pixels, we observe where each object was actually located (or whether an object was not located in any of the acquired pixels). This information is used to update the motion models.
4. If we allow for nonstationary object dynamics then we must now update our belief about the motion models to reflect that they may have changed.

Algorithm 1 Input: M, the set of all possible motion models, $I_1, ..., I_N$, a set of N frames, $f_{x_1}(x)$, the distribution of the object's location in the first frame. Output: $f_M(m)$, the probability that the object follows motion model m.

```
initialize f_M(m), the prior likelihood of each motion model
for t = 1 to N do
    compute f_{x_{t+1}}(x|m) for all m ∈ M
    f_{x_{t+1}}(x) = ∫_M f_{x_{t+1}}(x|m) f_M(m) dm
    choose set of pixels D to maximize ∫_D f_{x_{t+1}}(x) dx
    acquire pixels in D in I_{t+1}
    if object found then
        f_{x_{t+1}}(x) = 1 at object location, 0 elsewhere
    else
        f_{x_{t+1}}(x) = 0 for all x ∈ D
    end if
    f_M(m) = Pr(x_{t+1}|m) f_M(m) for all m ∈ M
end for
return f_M(m)
```

4. CASE STUDY

We now explain these steps in more detail with a simple case study. We assume that we are only acquiring a single object, and that the motion is known in advance to be the random walk model. Hence, the only unknown variable is the covariance. For the sake of clear diagrams, we generally assume that the object moves only in one dimension, meaning that the covariance consists of a single variable.

4.1. Initial Conditions

We assume the initial position of the object is known. As stated above, the only unknown variable is the variance of the object's motion, or, equivalently, the standard deviation σ. Our initial assumption is that this standard deviation lies between 0 and 10 with equal probability. The true σ (initially unknown by our system) is set to 1.

4.2. Prediction of the Distribution of the Object

Because the object follows a random walk, its expected location in the subsequent frame is the same as in the current frame, but it is still subject to the Gaussian state noise from (1). The problem is that the system does not know the standard deviation of this Gaussian, which would be required to compute the distribution of the object's expected location. However, the system does maintain the probability of any given standard deviation being the true standard deviation. Equation (5) shows how we compute the distribution of the object's position, $f_x(x)$, when the standard deviation σ is only known as a probability distribution $f_\sigma(\sigma)$. $f_x(x|\sigma)$ refers to the expected object distribution when σ is known, and is thus simply a Gaussian of mean 0 and standard deviation σ;

$$f_x(x) = \int_\sigma f_x(x|\sigma) f_\sigma(\sigma) d\sigma. \tag{5}$$

Figure 1 shows how the object's expected distribution changes as the model is learned.

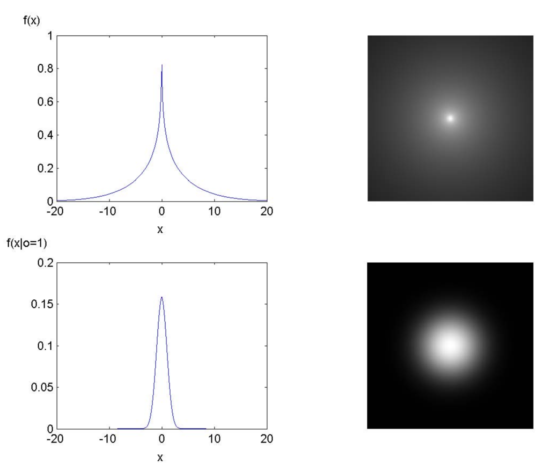

Fig. 1. The top row shows the expected object distribution when the standard deviation is unknown. The plots are in 1D (left) and 2D (right). The bottom row shows the distribution when the standard deviation is known to be 1.

Figure 2 shows the first four iterations of the algorithm. We see that as the knowledge of σ becomes more precise, the distribution of the object's expected location also becomes more precise.

The assumption thus far is that the object's original position is known. However, sometimes we fail to acquire the object in one frame, but must still predict its location in the next frame. Figure 3 shows an example where the object is not found in the current frame, and thus its location is only known probabilistically. The subsequent object distribution is the convolution of this current distribution and the random walk function.

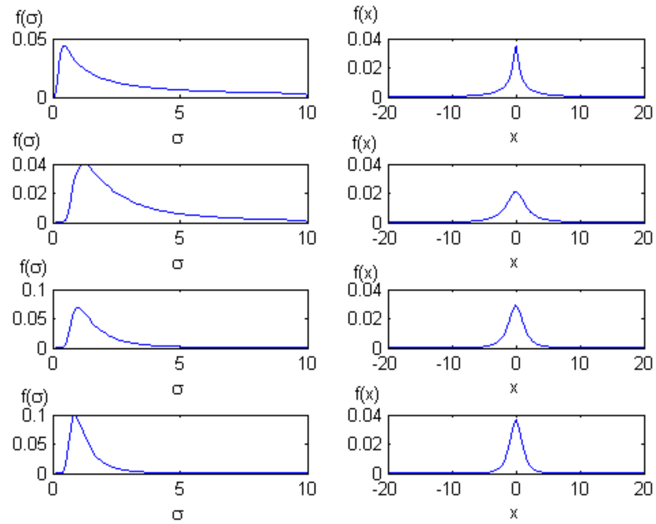

Fig. 2. Four iterations of the algorithm. The left side reflects our knowledge of σ; the right side shows the expected object distribution.

4.3. Acquisition of Pixels According to Policy

Suppose that we wish to acquire X pixels in each frame. In this case we acquire the X pixels with the highest probability of containing the object. It can be shown that this is also the set of pixels that maximizes the rate at which we learn the object's motion model.

4.4. Update of the Motion Model

On the basis of this acquisition, we now update our estimate of σ as follows:

$$f_{new}(\sigma) = \frac{\Pr(d \in D|\sigma) f_{old}(\sigma)}{\Pr(d \in D)}.$$

In this equation, D refers to the displacement of the object between the two frames. If the object was acquired in both frames then D is a single number representing the actual measured displacement of the object. However, if the object was not acquired, then the actual displacement of the object is unknown and thus D is the set of all possible values of displacement.

5. HIGHER-ORDER MODELS

When we consider higher-order models, we must also maintain distributions of the objects' past positions. In the second-order linear extrapolation, with state transition matrix F_{SLE}, the update equation for a single dimension is given as:

$$x_{t+1} = 3x_t - 3x_{t-1} + x_{t-2} + N(0, \sigma).$$

Thus, we need to maintain estimates of x_t, x_{t-1} and x_{t-2}. For the frames when the object is observed, this estimate will be an exact point. However, for frames when the object was missed, the position of the object must be represented probabilistically.

This added complexity means that it takes longer to learn σ than for the random walk model. Figure 4 shows the rate at which σ is learnt for each of the three motion models, showing that the lower-order models are faster to converge to the true value of σ.

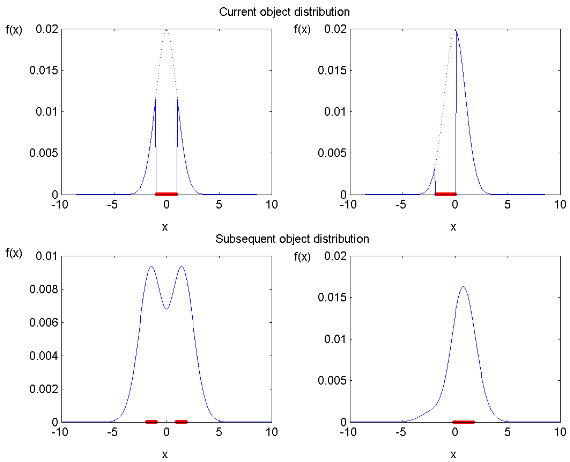

Fig. 3. The top row shows two example acquisition regions that failed to find the object, meaning that the object's location is only known probabilistically. The bottom row shows the resulting expected object distribution in the subsequent frame. The acquisition regions are marked in red along the x axis.

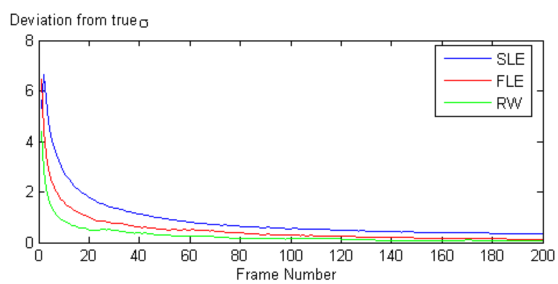

Fig. 4. The rate at which σ is learnt under each of the motion models.

Even once σ has been learnt, the higher-order models still require a larger average acquisition region. This is because when an object is missed in any given frame, the resulting uncertainty about the object's motion is propagated for longer when using a higher-order model. If the object must be captured in 90% of frames in a 1-dimensional setting, then an RW model requires an average acquisition region of length 3.78σ, an FLE model requires an average acquisition region of length 4.48σ, and the SLE requires 6.97σ.

6. DISCUSSION

The framework presented is very general and can be adapted to a wide variety of applications. Extension to multiple objects and more complicated motion models is simple and intuitive. The cost function can also be broadened to include the time taken to move the beam from one region of the image to another, rather than assuming a cost directly proportional to the number of pixels acquired. We can also adjust the goal that the system tries to acheive. In the case study presented, the goal is simply to acquire the object in every frame. Alternative goals are to learn the motion model as quickly as possible, or to know the locations of the objects in every frame as accurately as possible (but without necessarily acuqiring them). Although these goals are closely related, they result in slightly different acquisition strategies.

There is also potential to improve efficiency by using less greedy regions. Currently we choose the region that we expect to give the most information in the subsequent frame $t + 1$. However, if we know that we are going to acquire for at least the next N frames, we can sacrifice information in the immediately subsequent frame in return for more information in these future frames. For example, in the right side of Figure 3, we choose an initial region that is less likely to capture the object than the greedy region chosen in the left side of Figure 3. Although this region reduces detection probability in frame $t+1$, it actually increases the detection probability in frame $t + 2$. This occurs because even when the object is not captured in frame $t+1$, there is a higher degree of certainty about its position in frame $t + 2$.

Hence, there are many enhancements that are possible under the outlined framework. The limiting factor in these more advanced systems is the increased computational complexity. The main challenge of future work will be to ascertain what simplifications can be made to maintain a manageable computational load yet without adversely affecting system behavior.

7. ACKNOWLEDGEMENTS

The authors thank Estelle Glory for helpful insights.

8. REFERENCES

[1] S. Inoue, *Handbook of Biological Confocal Microscopy*, chapter Foundations of Confocal Scanned Imaging in Light Microscopy, 2005.

[2] K. König, *Handbook of Biological Confocal Microscopy*, chapter Cell Damage During Multi-Photon Microscopy, 2005.

[3] A. Diaspro, G. Chirico, C. Usai, P. Ramoino, and J. Dobrucki, *Handbook of Biological Confocal Microscopy*, chapter Photobleaching, 2005.

[4] T. E. Merryman and J. Kovačević, "Adaptive multiresolution acquisition of fluorescence microscopy data sets," *IEEE Trans. Image Proc., sp. iss. Molecular and Cellular Bioimaging*, vol. 14, no. 9, pp. 1246–1253, Sep. 2005.

[5] A. Genovesio, T. Liedl, V. Emiliani, W. I. Parak, M. Coppey-Moisan, and J.-C. Olivo-Marin, "Multiple particle tracking in 3-d+t microscopy: method and application to the tracking of endocytosed quantum dots," *IEEE Trans. Image Proc.*, vol. 15, no. 5, pp. 1062–1070, May 2006.

[6] A. Genovesio and J.-C. Olivo-Marin, "Tracking fluorescent spots in biological video microscopy," *Proc. SPIE Conf. 3D and Multidimensional Microscopy: Image Acq. and Proc.*, vol. 4964, pp. 98–105, May 2003.