Intelligent Acquisition and Learning of Fluorescence Microscope Data Models

Charles Jackson, *Student Member, IEEE*, Robert F. Murphy, *Senior Member, IEEE*, and Jelena Kovačević, *Fellow, IEEE*

***Abstract*—We propose a mathematical framework and algorithms both to build accurate models of fluorescence microscope time series, as well as to design intelligent acquisition systems based on these models. Model building allows the information contained in the 2-D and 3-D time series to be presented in a more useful and concise form than the raw image data. This is particularly relevant as the trend in biology tends more and more towards high-throughput applications, and the resulting increase in the amount of acquired image data makes visual inspection impractical. The intelligent acquisition system uses an active learning approach to choose the acquisition regions that let us build our model most efficiently, resulting in a shorter acquisition time, as well as a reduction of the amount of photobleaching and phototoxicity incurred during acquisition. We validate our methodology by modeling object motion within a cell. For intelligent acquisition, we propose a set of algorithms to evaluate the information contained in a given acquisition region, as well as the costs associated with acquiring this region in terms of the resulting photobleaching and phototoxicity and the amount of time taken for acquisition. We use these algorithms to determine an acquisition strategy: where and when to acquire, as well as when to stop acquiring. Results, both on synthetic as well as real data, demonstrate accurate model building and large efficiency gains during acquisition.**

***Index Terms*—Active learning, fluorescence microscopy, image modeling, intelligent acquisition, particle filter.**

I. Introduction

FLUORESCENCE microscopy is a powerful technique for live-cell imaging [1]. As the trend in biology tends more and more towards high-throughput applications, the amount of image data acquired with this technique is growing rapidly. One implication of this growth is that the limitations of the acquisition process, such as photobleaching, phototoxicity, and finite resolution, become increasingly important. A second implication is that visual inspection of the data becomes impractical, motivating the goal of automated image analysis. This paper presents a framework and algorithms to tackle both of these goals simultaneously.

Manuscript received September 08, 2008; revised May 13, 2009. First published June 05, 2009; current version published August 14, 2009. This work was supported in part by the National Science Foundation under Grant EF-0331657 and in part by the PA State Tobacco Settlement, Kamlet-Smith Bioinformatics Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Erik H. W. Meijering.

C. Jackson is with the Department of Biomedical Engineering and the Center for Bioimage Informatics, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

R. F. Murphy is with the Departments of Biological Sciences, Machine Learning, and Biomedical Engineering, and the Center for Bioimage Informatics and the Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

J. Kovačević is with the Departments of Biomedical Engineering and Electrical and Computer Engineering, the Center for Bioimage Informatics, and the Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: jelenak@cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2009.2024580

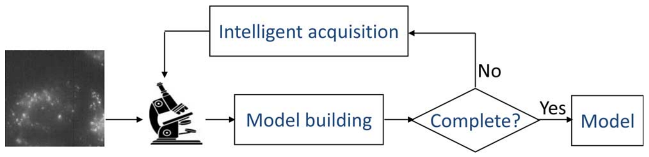


Fig. 1. Diagram of the proposed research. The model building module constructs a model from the microscope data, and the intelligent acquisition module determines which acquisitions to make to efficiently improve on this model.

A. Goal

We aim to develop a mathematical framework and algorithms to build accurate models of fluorescence microscope time series as well as to design an intelligent acquisition system based on these models. Fig. 1 gives a high-level view of our system. The model building module uses the input from the microscope to estimate a model that captures the essential information from the time series. The intelligent acquisition module uses the current estimate of the model to choose the data acquisitions that will improve on the model estimate as efficiently as possible. This process continues until the model has been learned with sufficient confidence, or until it cannot be further improved (for example, photobleaching has destroyed the fluorescent signal).

B. Motivation

Visual inspection of microscope images has two major drawbacks: it is time-consuming and inconsistent. These drawbacks have resulted in an increasing focus on developing fast and automated image analysis techniques to summarize image data in the form of a model. For example, an automated system to learn models of subcellular organization directly from images has been described [2]. Because the end result of acquisition is a model rather than an image, we can perform our analysis directly on the raw microscope data without the intermediary stage of a viewable image (see Fig. 1). This could give a more accurate result, as well as allow for a more flexible and optimized acquisition process, increasing the potential for intelligent acquisition.

The first motivation for intelligent acquisition is to reduce two major limitations of the acquisition process: photobleaching and phototoxicity [3]. In fluorescence microscopy, we label objects of interest in a specimen with fluorescent components called fluorophores. Under excitation light of a certain wavelength,

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

these fluorophores absorb energy and subsequently emit light at a longer wavelength. This emission light is filtered from the excitation light prior to detection, revealing the locations of the fluorophores and the objects of interest. Photobleaching is a mechanism whereby, after prolonged exposure to excitation light, a fluorophore can move into an excited triplet state and undergo a covalent modification that destroys its ability to fluoresce. Phototoxicity occurs when a fluorophore in the excited triplet state undergoes a reaction with molecular oxygen, releasing a free radical. These free radicals can cause cell damage and eventually cell death. Because we cannot acquire data without causing these effects, there is motivation to reduce total data acquisition in a way that does not sacrifice essential information about the specimen. The model building module defines a model space that concisely captures this essential information, allowing the intelligent acquisition module to determine where to acquire, when to acquire, and at what resolution to acquire, to build the model as efficiently as possible. As a result, in our framework, both photobleaching and phototoxicity are reduced.

The second motivation for intelligent acquisition is to reduce total acquisition time. The first way to do this is by automatically determining when to stop acquiring. This avoids the problem of acquiring more frames than necessary (which wastes time and causes unnecessary photobleaching), and also removes the risk of acquiring too few frames and, thus, not obtaining the desired information. The second way to acquire information faster is by choosing the optimal temporal and spatial resolution. Our work is primarily aimed at confocal microscopy used to acquire high-resolution 3-D time series. In confocal microscopy, cross-sectional slices at different heights of the specimen are imaged by shifting the microscope's focal plane. A slice is acquired by illuminating each pixel in the xy-plane in turn, and using a small pinhole to ensure that only fluorescence from the focal plane reaches the detector. Thus, images are captured as a set of pixels, resulting in a trade-off between temporal and spatial resolution. With intelligent acquisition, we can choose the optimal trade-off between temporal and spatial resolution, along with the best region to acquire, to build an accurate model as quickly as possible.

C. Related Work

Intelligent acquisition for fluorescence microscopy is a recent problem. Related work on efficient acquisition for fluorescence microscopy was done by Merryman & Kovačević [4]. They presented an algorithm to reduce the number of pixels acquired in a 2-D or 3-D image when using a laser scanning confocal microscope, with the end application being recognition of proteins based on their subcellular location [5]–[8]. The goal was to reduce the time spent acquiring low fluorescence regions, which presumably contain little useful information. The algorithm begins by scanning the field at 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. The limitation of this technique is that it cannot adapt to specifically seek out information required for the end application, nor can it use knowledge of the cellular dynamics.

Hoebe *et al.* introduced controlled light-exposure microscopy [9], which can use a different exposure time for each pixel. If no significant fluorescent signal is detected at a pixel, the exposure time for that pixel is reduced. Similarly, if the signal is very strong, the exposure time will also be reduced because the signal-to-noise ratio will still be high. As a result, this method allows images to be acquired faster and with less overall light exposure, thus reducing photobleaching and phototoxicity. This technique could be included within our proposed framework.

Work on efficient acquisition is also found in the field of magnetic resonance imaging (MRI). For example, Liang & Lauterbur [10] present a method to efficiently acquire a time series of images by observing that the high-resolution image morphology does not generally change from one image to another. Then, using a generalized series model, they eliminate the repeated encodings of this stationary information in the conventional Fourier methods. An alternative approach uses a singular value decomposition of the first (base) image to design excitation sequences that efficiently acquire the data in subsequent images [11]. A more comprehensive overview of efficient acquisition in MRI is given by Tsao *et al.* [12].

More generally, the problem of intelligent acquisition comes under the framework of active learning, which refers to any form of learning in which the learning program has some control over the inputs on which it trains [13]. Assuming that data acquisitions are expensive, the goal is to request the data that is most informative. Two of the early proposals for this were uncertainty sampling [14], which chooses acquisitions whose result the learner is least certain about, and query by committee [15], which chooses acquisitions that cause the most disagreement among the version space (the space of classifiers which are consistent with the previously labeled instances). A survey of more recent work is given by Anderson & Moore [16].

D. Roadmap

We start in Section II by providing the necessary background on the state-space approach and particle filters, which are used heavily for single object model building. In Section III, we present our overall framework for acquisition. Section IV discusses how the acquired data is used to build models. Section V outlines the algorithms we use for intelligent acquisition, and experimental results are given in Section VI.

Throughout this paper, we use *3-D image* to denote a static image in 3-D, where a 3-D image[1] is made up of a set of 2-D images at different heights in the specimen. Each of these 2-D images is called a *z-slice*, and, thus, the height is always denoted by the z-dimension. We use *3-D time series* to denote a sequence of 3-D images. Unless otherwise specified, all experiments and discussion refer to 3-D time series.

II. Background

In this section, we review the state-space approach upon which our proposed framework is based. We also review the object motion models that we use later in the paper. Finally, we look at particle filters, which are used in Section IV.

A. State-Space Approach

By using the state-space approach, we assume that the time series has a true underlying state, and that our observations give

[1]In biology, a 3-D image is often called a z-stack.

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

some interpretation of this state [17]. The state-space equations in their most general form are given by

$$\mathbf{x}_{t+1} = g(t, \mathbf{x}_t, \boldsymbol{\nu}_t) \quad (1)$$

$$\mathbf{z}_t = h(t, \mathbf{x}_t, \mathbf{w}_t) \quad (2)$$

where $\mathbf{x}_t$ is the state vector—the set of state variables that can represent the entire state of the system at time t. For example, if we were modeling objects within a cell, then the state vector could include the positions, shapes, and motion models of these objects. The vector $\mathbf{z}_t$ refers to the observed vector. The first (1), is the state update equation, with the function g describing how the state of the system evolves with time. This includes a noise term $\boldsymbol{\nu}_t$ to reflect that the model of the state evolution will not be exact, and that the actual state will differ from the predicted state. In the measurement (2), h relates the system state to the observed measurement at time t, with $\mathbf{w}_t$ allowing for measurement noise.

The reader may be more familiar with the linear forms of these equations

$$\mathbf{x}_{t+1} = \mathbf{G}_t\mathbf{x}_t + \boldsymbol{\nu}_t \quad (3)$$

$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{w}_t \quad (4)$$

which can be used when the dynamics of the system are known and linear. However, in our case, the system dynamics are generally unknown—the point of the model building process is generally to learn these dynamics. Because we want to view g as a constant and known function, we follow the approach of Berzuini *et al.* [18], and assume a very generic model for the system. The unknown parameters of this generic model are then placed as static variables in the state vector $\mathbf{x}_t$. Thus, learning the static variables in $\mathbf{x}_t$ is equivalent to building a specific model of the system. This allows us to view g as a constant and known function but also forces us to use the nonlinear form of the state update equation (1).

Our intelligent acquisition process often involves acquiring only within a certain region of an image, meaning that objects are only observed if they lie within this region. As a result, our measurement function h is also nonlinear and we must use the nonlinear form of the measurement equation (2). Note that h is a variable function controlled by the intelligent acquisition module. This distinguishes our application from most state-space models.

B. Motion Models

Because much of this paper focuses on learning the motion models of individual objects within cells, we present the motion models that we use for single object modeling. This section assumes 3-D time series. There are two motion models that subcellular objects are commonly observed to follow: random walk (RW) and constant velocity (CV).

1) Random Walk Motion Model: In the RW model, objects move in a random direction between frames. The position of an object in a frame depends only on its previous position, and, thus, velocity and acceleration of the object are not conserved. Its position at time t, $\mathbf{y}_t$, is simply its previous position, $\mathbf{y}_{t-1}$, perturbed by the displacement $\mathbf{d}_t$ of additive Gaussian noise of mean $\mathbf{0}$ and covariance $\boldsymbol{\Sigma}$, $N(\mathbf{0}, \boldsymbol{\Sigma})$. The parameters of the covariance matrix $\boldsymbol{\Sigma}$ are often known as rate parameters because they govern the rate at which the objects move in each dimension. The RW model is thus described by

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \mathbf{d}_t. \quad (5)$$

2) Constant Velocity Motion Model: In the CV model, objects move with a constant velocity $\mathbf{v}$ between frames. Once again, we have displacement $\mathbf{d}_t$ as additive Gaussian noise, which determines the extent to which velocity is conserved. The CV model is governed by

$$\mathbf{y}_t = \mathbf{y}_{t-1} + \mathbf{v} + \mathbf{d}_t. \quad (6)$$

We can see from this equation that the RW model is a special case of the CV model, with $\mathbf{v} = \mathbf{0}$. Therefore, both of these models can be characterized by $(\mathbf{v}, \boldsymbol{\Sigma})$, where $\mathbf{v}$ is a 3-D vector. Note that a general symmetric covariance matrix $\boldsymbol{\Sigma}$ has $D(D+1)/2$ independent parameters [19], where D is the dimension, and, thus, for $D = 3$, we have six independent parameters. Combining this with the three parameters of $\mathbf{v}$, the model has nine independent parameters in total.

C. Particle Filters

Because both the state update (1), and the measurement (2), are nonlinear, we often use a particle filter for likelihood estimation. This is a simulation-based technique that requires neither linearity nor Gaussian noise, and its performance often exceeds that of the extended Kalman filter (a nonlinear version of the Kalman filter [20]). Particle filters are also known as sequential Monte Carlo methods [21].

The basic approach in particle filtering is to represent the posterior distribution of the state vector using a set of sample states known as particles. We begin by generating N particles drawn from the prior distribution of the state space. At each time step, we propagate the particles forward according to the state update (1). As observations are made, those particles consistent with the observations are given high weighting, and inconsistent particles are given low weighting. To avoid a concentration of particles in low-likelihood regions, a resampling procedure then duplicates those particles with a high weighting and eliminates those with a low weighting. There are many ways of doing this [18], [22], [23]; we use sampling-importance-resampling [20]. When an observation is made, we assign each particle a weight according to the posterior probability of the particle given the observation. We then resample the particles with probabilities proportional to their weights—that is, we draw N particles with replacement from the current particle set.

The main limitation of particle filters is that high accuracy requires high computation time. This is especially true when the problem is high-dimensional or when there are unknown static parameters in the state space. Because the latter situation occurs frequently in our work, we draw special attention to an enhancement given by Gordon *et al.* [20], known as roughening, which treats these unknown static parameters as though they were dynamic, by adding a small amount of random noise to them at each iteration of the algorithm. The random noise makes some particles move closer to the truth, and others further away. Those that move closer are more likely to agree with future observations and, thus, more likely to be duplicated. Hence, the particles eventually converge on the true parameter values.

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

Particle filters are attractive because they make few assumptions about the dynamics of the state space model and they are easy to implement even for complex models [24]. We do not need to derive equations to calculate the model likelihood for any given observation. Instead, we only need to simulate multiple instances of particles following different models, and the likelihood function reveals itself.

III. Proposed Framework

Fig. 1 gives a high-level diagram of our framework with two main modules: model building (M) and intelligent acquisition (A). The M module uses the acquired data, along with any prior knowledge about the time series, to build meaningful models. For this, we must define the model space, which is the set of all possible models that we wish to consider. We can also define a prior likelihood on each of these models. The A module determines what data is the most useful for building these meaningful models, and instructs the microscope to acquire accordingly. This module also determines when to stop acquisition.

The M module impacts the A module in two ways: First, the type of models that we consider dictates what information is important, enabling more efficient acquisition. For example, if we were considering models solely related to the vesicles within a cell, the A module could ignore regions not containing vesicles. Second, the current estimate of the model is used to guide future acquisitions. For example, if we only wish to acquire regions with vesicles, the current estimate of the model is used to predict where those regions will be. The feedback from the M module to the A module is represented by the loop in Fig. 1: the current model estimate is used to determine future acquisitions, which in turn refine the estimate of the model.

When choosing the model space, we must ensure that it captures all information from the time series that is required for the end application. However, to ensure that the A module does not acquire data unnecessarily, we must also exclude any information that is not required for the end application. In general, this is achieved by describing models that contain as few parameters as possible.

A. Modeling

The focus of this paper is to model the motion of objects in a time series. We do not try to learn the actual trajectory of an object, but rather just its dynamics. The dynamics of an individual object can be described using the motion models outlined in Section II-B.

For a cell containing many objects, we do not usually need to learn the dynamics of every individual object. Instead, we are interested in the motion types present in the time series, and the proportion of objects in each motion type. For example, we may wish to identify that 60% of the objects move under an RW model of certain parameters, and the remaining 40% move under a CV model of certain parameters. Scenarios such as these, which are relevant in practice, have a high potential for intelligent acquisition because, depending on the level of detail required, the number of parameters to learn may be relatively small and can, thus, be learned with a fairly small subset of the total data available.

At this stage, we have primarily developed algorithms for the special case of a single object moving in a cell. We verify the effectiveness of these algorithms in Section VI, and these experiments provide a proof-of-concept that efficiency gains are possible. However, recognizing the importance of real data validation, we also present some simple algorithms designed for multiple object scenarios, and these are tested on real data. These algorithms assume that all objects fall under the same motion type, and, thus, the model learned is the one that is the best overall fit to the objects present in the time series. We anticipate extension to multiple motion types to be rather straightforward.

Although we have chosen to validate the idea using object motion modeling, our framework (Fig. 1) is more general and applies to other scenarios as well. A simple example was already mentioned in Section I-C where Merryman and Kovačević [4] proposed an intelligent acquisition strategy for the purposes of classification. In that example, the model is the classifier output.

IV. Model Building

The goal of model building is to use the raw data acquired from the microscope, along with any prior knowledge about the time series, to construct a model. Data from the microscope could come in several forms—from any subset of pixels, at any arbitary resolution, and at any frame rate. In this section, we look at both single object models and multiple object models. We present model building for single object models in the most general case, but for multiple object model building we require a constant frame rate and a value for every pixel. This restricts the intelligent acquisition algorithms available for the multiple object case, but, as we see in Section V, still leaves scope for several methods.

Throughout this section, we make the following assumptions. 1) We treat the object as a point source, meaning that we ignore its shape and size. 2) We assume perfect object detection provided that the appropriate region of the image is acquired. 3) We assume that all pixels in a frame are acquired at precisely the same instant.

A. Single Object Model

In the simple case where every pixel and every frame are acquired, and where we restrict ourselves to the Gaussian models proposed in Section II-B, model building for a single object is easy: We record the displacement of the object in each frame, and use the sample mean and covariance of these displacements as our model parameters. Even with the small error introduced by pixelation, this closed-form method is fast and accurate.

However, this simple case is restrictive for two reasons. First, we want the intelligent acquisition module to be able to choose only a subset of pixels in each frame, and to sometimes skip frames entirely. Second, even the simple Gaussian models still require some modification to stop objects from crossing the cell boundary, and in future we may want to test different motion models entirely. For these reasons, we present particle filters as our primary method for single object model building, but noting that an analytic closed-form solution is faster when available.

To use particle filters, we must first choose a set of models. As described in Section II-B, a motion model consists of $(\mathbf{v}, \mathbf{\Sigma})$, where $\mathbf{v} = \mathbf{0}$ for an RW model. We initialize the particle filter

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

by generating N particles from the model space. Each particle is assigned values for $(\mathbf{v}, \mathbf{\Sigma})$ with probabilities proportional to the joint prior likelihood of the parameters. If no prior information is available then we assume that RW models and CV models are equally likely, and that all parameter values within these models are equally likely (with some upper bound). Note, however, that $\mathbf{\Sigma}$ must be symmetric positive semi-definite to be a valid covariance matrix. We must also assign initial positions to each of the particles. We assume that we begin acquisition by acquiring a complete frame and that we observe the true position of the object. Thus, the initial positions of the particles can be set to the actual initial position of the object.

Following an observation of the object, we use the following simple weighting procedure: If a particle is at the same pixel as the observed object, we keep it; if a particle is at a different pixel, we eliminate it (assign it zero weight). If we acquire a set of pixels and do not observe the object, then we eliminate the particles in the acquired set of pixels, and retain all other particles. In either case, the surviving particles all have equal weight, and so the resampling procedure just involves duplicating these particles to keep the overall number at N. We could improve efficiency and reduce degeneracy by giving fractional weight to particles that are in neighboring pixels of the observed object. However, this complicates the intelligent acquisition algorithms, so we leave it for future work.

Using this procedure, the likelihood of any model is given by the distribution of the models of the surviving particles. As N approaches infinity, this distribution converges on the true likelihood function. Furthermore, as the number of observations increases, the likelihood function converges to the object's true model.

1) Computation Time: In practice, we can only simulate a finite number of particles, and, thus, we have a trade-off between accuracy and computational efficiency. The number of particles required depends on the predictability of the object's motion and the distance it tends to move between frames. If objects move by 3–5 pixels per frame on average, then we found that at least 5000 particles were needed to avoid degeneracy problems, but up to 100,000 particles is preferable (beyond this, we did not observe much benefit). Even with 100,000 particles, computation time is under 0.1 s per frame on an Intel Core Duo 2.2-GHz processor with 1.96 GB of memory (all times reported henceforth are for this configuration). Our experiments are done with a frame size of $1000 \times 1000 \times 15$, but the number of pixels in the frame has a negligible effect on computation time in comparison to the number of particles.

Note that the accuracy only needs to be high enough to determine an acquisition strategy. If necessary, when acquisition is complete, we can go back and build a more accurate model (with more particles, or using a completely different method).

B. Multiple Object Model

Although the single object method could be extended to multiple objects by implementing a separate particle filter for each object, this introduces the issue of object correspondence. We would need to know which objects in frame t correspond to which objects in frame $t+1$, which becomes hard if only a subset of the pixels are known in each frame. We would also need to model the likelihood of an error in these correspondences, which will depend on the time elapsed between frames, making the analysis difficult in a variable frame rate scenario. Because of these difficulties, we leave a fully developed model building module for future work. However, recognizing the need to show experiments on real data (which contain multiple objects), we now present a model building algorithm for the simplified case in which the frame rate is constant and in which we have the full set of pixels. Note that although the algorithm requires the full set of pixels as input, we do not necessarily have to acquire every pixel. We could still acquire only a subset of the pixels provided that we can confidently estimate the values for those left unacquired. Additionally, as mentioned in Section III-A, we assume that all objects move under the same model.

We could still use a particle filter but this is unnecessary in this simplified case. Instead, much as for the simple case in single object modeling, our basic strategy is to record all the displacements of every object in every frame, and then use the sample mean and covariance of these displacements as our model parameters. The problem is that without object correspondence data, these displacements are unknown. Our strategy is to first define a window that describes the maximum distance an object can move between frames. For example, when testing on real data in Section VI-B, we assume that an object cannot move more than 32 pixels in the x- or y-direction, and not more than 2 pixels in the z-direction. Then, for every object in every frame, we record all possible displacements of that object, giving them equal weight if there are several possibilities. Following that, we learn a model from the set of displacements and their weights. The final stage is to iterate between using the model to update the weights associated with each possible displacement, and recomputing the model based on these updated weights, until convergence. This process is also shown in Algorithm 1.

Algorithm 1 *Input*: locations of objects in frames $1 \cdots t$. *Output*: $m()$, the model, a function giving the likelihood of each displacement

```
for all objects $o$ in frames $1 \cdots (t-1)$ do
    set $D_o$ to the set of possible displacements for $o$
    initialize $W_o$ to give constant weight to each
    displacement in $D_o$
end for
repeat
    learn $m()$ from (D,W)
    for all objects $o$ in frames $1 \cdots (t-1)$ do
        for all displacements $d$ in $D_o$ do
            set $W_o(d)$ to $m(d)$
            normalize $W_o$
        end for
    end for
until $m()$ no longer changing
return $m()$
```

The only remaining detail is to specify which model we use to describe the observed displacements. One choice is to use the models from Section II-B, in which case we define a model, m, as $m = (\mathbf{v}, \mathbf{\Sigma})$. We can then estimate $(\mathbf{v}, \mathbf{\Sigma})$ simply by finding

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

the sample mean and covariance of the set of weighted displacements. However, with real data, we make one small enhancement to this model: In real data, objects are often observed to appear or disappear (whether due to imperfect detection processes or some other mechanism). When objects appear, they generally appear in the vicinity of other objects, and are often confused with the displacement from an already existing object. As a result, the observed displacements are better modeled by $m = N(\mathbf{v}, \mathbf{\Sigma}) + c$, where c is constant within the window mentioned above, and 0 outside of this window. The values of $(\mathbf{v}, \mathbf{\Sigma}, c)$ can be learned from the weighted displacements using the expectation-maximization algorithm. We still describe our final model using only $(\mathbf{v}, \mathbf{\Sigma})$ because these are the parameters that describe the objects' motion and that is what we are interested in. However, allowing for the constant c during the estimation procedure gives more accurate values of $(\mathbf{v}, \mathbf{\Sigma})$. This can be proven using our model verification method that is described in Section IV-C.

1) Computation Time: The computation time depends mainly on the number of objects present. In Section VI-B, we test this algorithm on real data. These data sets contain up to 1000 objects in a frame. Our method processed a frame in about 0.05 s, whereas the acquisition time for these frames was about 45 s. Therefore, the time taken to build a model is negligble in comparison to the acquisition time.

C. Model Verification

For simulated data, we can directly test whether we are learning the correct model, because we know the ground truth. For real data, we instead take an indirect approach to verify a model. Our method is as follows: prior to acquiring a frame, we use our model to estimate the probability of any pixel in that frame containing an object. Once we have acquired the frame and observed the actual locations of the objects, we then compute the likelihood of those observations given the prediction of our model.

This likelihood value is not meaningful by itself. However, we can then compare it to the likelihoods that we would obtain using alternative models. We demonstrate this in Section VI-B1, where we show that a model built using Algorithm 1 gives a higher likelihood than that given by a naive constant model, a nonadaptive Gaussian model, or a model trained on fewer frames.

V. Intelligent Acquisition

The model building section outlined how to build the best model from the model space to describe the time series. Now we discuss the data acquisitions that allow us to build this model at the lowest cost. The cost of learning a model is a function of the time taken to learn it, and the phototoxicity and photobleaching incurred during acquisition. When searching for the optimal acquisition strategy, we can choose where to acquire, when to acquire, and when to stop acquiring. In addition to minimizing cost, we also need to satisfy physical constraints. For example, because it takes finite time to acquire each pixel, there is a trade-off between spatial resolution and temporal resolution.

Our proposed method associates a reward and a cost with each acquisition. We then aim to maximize the reward relative to the cost. In this section, we begin by defining how to evaluate the error in the model. Next, we examine how to estimate the cost and the reward associated with any acquisition. Finally, we describe how to use this information to determine where to acquire, when to acquire, and when to stop acquiring.

A. Model Evaluation

To test the effectiveness of our algorithms, we must introduce a way of evaluating the error between our predicted model and the true model. To do this, we define a distance function $d(m_1, m_2)$ between two possible models m_1 and m_2. Recalling that a model consists of $m = (\mathbf{v}, \mathbf{\Sigma})$, we define this distance as follows:

$$d(m_1, m_2) = \|\mathbf{v}_1 - \mathbf{v}_2\|_2^2 + \|\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2\|_1 \tag{7}$$

where the first term is squared to ensure that it is of the same dimensionality as the second. The choice of distance function is not crucial, and functions such as the Kullback-Leibler divergence give similar qualitative results. The function chosen in (7) is convenient because it allows a fast method to evaluate the reward of an acquisition in Section V-C.

For the case where the M module outputs a point estimate of the model, as it does for multiple object models (see Section IV-B), we simply define the error in the model estimate m as $d(m, m_t)$, where m_t is the true model. When the M module outputs a set of particles instead of a point estimate, we have two approaches. One approach would be to simply form a point estimate and use the same method. However, a cleaner method is to take the mean distance between the models of all the surviving particles $(m_1 \dots m_N)$ and the true model m_t

$$E = \frac{\sum_{i=1}^{N} d(m_i, m_t)}{N} \tag{8}$$

where $d(m_i, m_t)$ is the distance between the model governing particle i and the true model m_t, and where there are N particles in total. Note that this function is invariant to the number of particles. For example, if we increased N to kN by duplicating each particle k times, E would remain unchanged. Also, for simplicity, (8) assumes that every particle has equal weight, but it is trivial to extend it to weighted particles.

B. Cost Evaluation

We now look at the cost associated with an acquisition. Our overall goals are to reduce the amount of acquisition time required to build a model, and to reduce the photobleaching incurred during this process. Our cost function reflects these two goals. In 3-D imaging, photobleaching occurs every time an object is acquired. Additionally, with a laser scanning confocal microscope, acquisition of a pixel will also cause photobleaching of similar magnitude for any objects in the out-of-focus planes. Hence, photobleaching is proportional to the number of times any pixel with the same xy-coordinates of an object is acquired, regardless of the z-coordinate of that pixel (this assumes constant illumination intensity and exposure time). Note that if we were to lower the illumination intensity or shorten the exposure time then we would reduce photobleaching. Adjusting these quantities could form part of the intelligent acquisition module. However, in our experiments, we assume constant illumination

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

intensity and exposure time because we do not have a good model of how varying these quantities would affect the quality of the image.

We do not define a generic phototoxicity model because it depends on the type of cells being imaged, and furthermore, phototoxicity is harder to measure directly. For the purposes of our simulations, we simply assume that the phototoxicity cost is proportional to the photobleaching cost.

Defining $Z(p)$ as the set of pixels with the same xy-coordinates as pixel p, and N_p as the number of particles in p, the cost of a frame acquisition is given by

$$C = \tau t + \rho \sum_{p \in P} \frac{\sum_{p' \in Z(p)} N_{p'}}{N}. \tag{9}$$

In this equation, the first term reflects the time cost: t is the elapsed time since the last acquisition, and τ is the cost per unit time. If τ is high then the system will try to minimize cost by finishing acquisition as quickly as possible. The second term is proportional to the overall probability of detecting an object using the set of pixels P, combined with the probability of acquiring any pixel with the same xy-coordinates as the object. ρ represents the photobleaching cost associated with each exposure of the object, and so this second term reflects the expected photobleaching cost of the acquisition. The ratio τ/ρ determines the relative importance of minimizing acquisition time versus minimizing photobleaching.

C. Reward Evaluation

The reward of an acquisition is the reduction in model error that results from that acquisition. In this section we show how to accurately predict this reduction for the case of a single object model. Because this method relies on having a full likelihood estimate of the model, it cannot yet be applied to multiple object scenarios (our model building method for multiple object scenarios only gives a point estimate of the model).

To estimate the expected reduction in error, we first estimate the current error in the model, and then predict what this error will be after making the acquisition. We cannot use (8) to compute the current error directly because (8) contains the true model, m_t, which is unknown. Instead, we assume that m_t is the model of one of the surviving particles, giving equal probability to all of them. With this assumption, we can recast (8) as follows:

$$E_c = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} d(m_i, m_j)}{N^2} \tag{10}$$

where E_c is the estimate of the current error, and $d(m_i, m_j)$ is the distance between the model governing particle i and the model governing particle j. Thus, E_c is equivalent to the expected distance between a randomly chosen pair of particles (with replacement).

We can also predict what the error will be after making the acquisition because (1) we know the probability with which an object will be found in any given pixel, and (2) we can estimate the resulting error if an object is found in any given pixel. The probability of the object appearing in pixel p is N_p/N. The resulting error if the object appears in p is given by (10), but with the summation applying only to the particles in this pixel. Thus, if P denotes the set of pixels in the frame, the predicted error after acquiring a frame, E_f, is given by

$$E_f = \sum_{p \in P} \frac{N_p}{N} \frac{\sum_{i=1}^{N_p} \sum_{j=1}^{N_p} d(m_i, m_j)}{{N_p}^2} \tag{11}$$

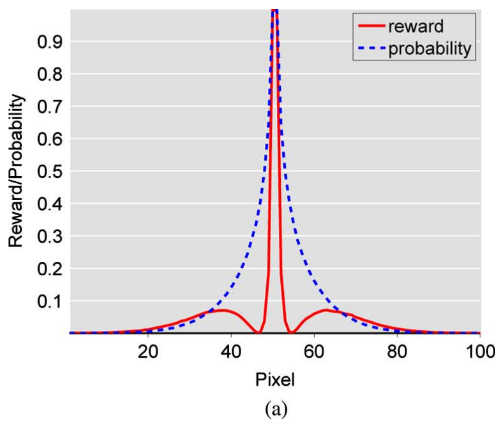
where m_i and m_j refer to the models of the particles present in pixel p. Using (10) and (11), we can estimate the reward of an acquisition as the estimated current error, E_c, minus the predicted future error, E_f. Thus, $R = E_c - E_f$.

1) Partial Frame Acquisitions: Equation (11) does not only apply to the case where we acquire every pixel in a frame. We can also use it to predict the resulting error when we acquire only some of the pixels in the frame. Suppose that we acquire only those pixels in the left half of the image: If the object lies in the left half, then we will find it and we will know which pixel it is in. If the object lies in the right half, then we will not find it, but we will know that it lies somewhere in the right half. Thus, we can view the entire right half of the image as being one big pixel, and still use (11). Similarly, to predict the resulting error when we acquire at different resolutions, we simply combine pixels together to match the appropriate resolution.

We may expect that the reward of acquiring a single pixel simply depends on the likelihood that that pixel contains the object. However, this is not the case. In Fig. 2, we simulate an object moving in 1-D under an RW model. We do this in 1-D solely for plotting purposes—a similar result can be shown in 3-D. Plot (a) shows the first frame of the simulation, where the solid red line plots the reward associated with acquiring a pixel, and the dashed blue line plots the probability of finding the object in the given pixel. Although these curves are close to each other, their shapes are clearly different. We can see that there are two distinct points where the reward curve drops to nearly zero while the probability curve remains relatively high. The reason for this is that, even if the object is found at these points, we gain very little information about the rate parameter, $\mathbf{\Sigma}$ (which is just a single number in the 1-D case). To understand this, Fig. 3 shows 10 Gaussian distributions with standard deviations between 1 and 3. We can see that all of these curves come very close at two distinct points, and, thus, observing such a displacement would give little information as to the true underlying Gaussian distribution.

In Fig. 2(b), we consider the ratio of reward to cost, which we define as the benefit. When learning the model quickly is of paramount importance (i.e., the time cost is high), the benefit curve follows that of the reward curve and is shown by the solid red line. However, when minimizing photobleaching is of paramount importance, the benefit curve instead follows that of the dotted black line. Here, we have assumed that the photobleaching cost of a pixel is proportional to the likelihood of that pixel containing the object. Although this is the curve when the time cost is zero, in practice, we must always assign some time cost to ensure that the model is learned in reasonable time. The dashed blue line shows the benefit curve when the time cost of one frame is 10% of the photobleaching cost of one exposure.

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.



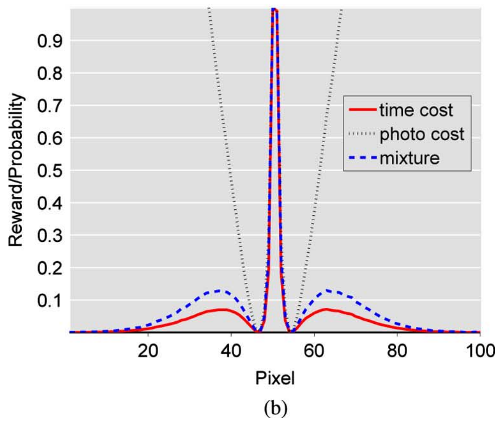


Fig. 2. Reward associated with a pixel acquisition. In this 1-D example, the object is assumed to move under an RW model. Plot (a) shows the first frame in the simulation. The dashed blue line shows the probability of finding the object in a given pixel. The solid red line shows the reward associated with acquiring this pixel. Plot (b) shows the ratio of reward to cost. The solid red line shows this ratio when we only consider the time cost. The dotted black line shows the ratio when we only consider the photobleaching cost. The dashed blue line shows the ratio when we consider both costs, with the time cost of one frame being 10% of the photobleaching cost of one exposure.

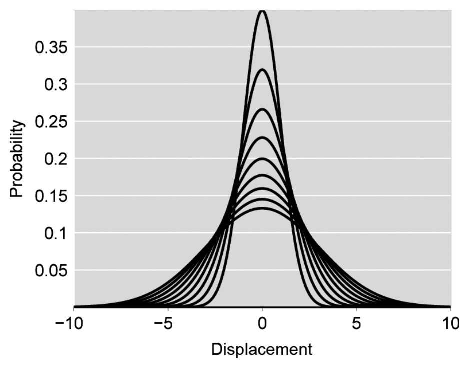


Fig. 3. Ten Gaussian probability distributions, each with a different variance. These curves all come close to two distinct points, meaning that a sample at one of these points would give little information as to which probability distribution the sample was drawn from.

D. Where to Acquire

In every frame, we must determine how many pixels to acquire, and which combination of pixels will be most beneficial. We discuss the single object case and multiple object case separately.

1) Single Object Model: As stated at the beginning of this section, our goal is to maximize the reward relative to the cost, which we define as the benefit. To do this, we take a greedy approach, and look for the combination of pixels that can maximize this benefit in the immediately subsequent frame. Note that there is no guarantee that this will maximize the benefit in the long-term.

Even maximizing the benefit in just the subsequent frame is a computationally daunting task. In any given frame of N pixels, there are 2^N combinations of pixels we could acquire. Clearly it is too resource-intensive to estimate the reward and cost for all of these combinations, and so instead we use a greedy procedure in which we continue adding the pixels with the highest marginal benefit to our set until the overall benefit stops increasing. Algorithm 2 describes this in more detail, and the effectiveness is verified experimentally in Section VI-A1.

Algorithm 2 *Input*: the set of particles in frame t. *Output*: P_{acq}, the set of pixels to acquire in frame t

```
$P_{acq} = \emptyset$
$B = 1$
**repeat**
    **for all** pixels $p$ in $\overline{P_{acq}}$ **do**
        set $R_p$ to reward of acquiring $\{P_{acq}, p\}$
        set $C_p$ to cost of acquiring $\{P_{acq}, p\}$
        $B_p = R_p / C_p$
    **end for**
    $p' = \arg\max_p B_p$
    **if** $B_{p'} >= B$ **then**
        $P_{acq} = \{P_{acq}, p'\}$
        $B = B_{p'}$
    **end if**
**until** $B_{p'} < B$
**return** $P_{acq}$
```

2) Multiple Object Model: Our model building method for multiple objects, outlined in Section IV-B, requires a full set of pixels as input (as opposed to the single object method, which only required a subset). However, we can still leave a pixel unacquired provided that we can confidently predict whether or not it will contain an object and can, thus, input its predicted value into the M module as though it had actually been acquired.

If we are almost certain that a pixel will contain an object, then clearly we reduce photobleaching by not acquiring this pixel. However, if objects move quickly and unpredictably, we can seldom be certain of a pixel containing an object. More commonly, we can only say with near certainty when a pixel will *not* contain an object. Although skipping pixels that we know will not contain objects does not actually reduce photobleaching in the focal plane, it does reduce photobleaching in the out-of-focus planes, because in 3-D imaging acquiring a pixel in one z-plane also causes photobleaching to any object with these xy-coordinates in another z-plane. Also note that by reducing the number of pixels acquired, we are increasing the rate

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

at which we can acquire a frame. This could allow for increased temporal resolution, and may also reduce any phototoxic effects that result from light exposure to other parts of the cell.

The trade-off with this method is deciding how certain we must be about a pixel's value to justify skipping it altogether. If we make this threshold too high then we end up acquiring too many pixels and do not significantly reduce photobleaching. If we make the threshold too low and acquire too few pixels, our assumption that the unacquired pixels do not contain objects will no longer hold, and the resulting model will be inaccurate. We show the effects of different thresholds experimentally in Section VI-B2.

E. When to Acquire

In addition to deciding where to acquire, we must also decide when to acquire. If we wait longer to acquire a new frame, more will have changed, which may mean that the acquisition will provide more information. This is beneficial, because to reduce photobleaching we want to get as much information as possible from each acquisition. However, in some cases, waiting too long to acquire a frame results in losing information that could best be gained with high temporal resolution. Furthermore, we then take longer to learn the model, and in the case of multiple objects, tracking performance is degraded.

Because the model building method for multiple objects (see Section IV-B) does not handle skipped frames, we focus solely on the single object case. To determine the best time to acquire, we use Algorithm 3. This algorithm first calculates the maximum benefit that we can achieve if we acquire in the next frame $t+1$, and then calculates the maximum benefit we can achieve if we only acquire in frame $t+2$. Recall that the benefit is defined as the ratio of reward to cost. The rewards can be calculated as described in Section V-C. The time cost of waiting until frame $t+2$ is twice that of acquiring in frame $t+1$. Assuming all pixels in the frame are acquired, the photobleaching cost will be the same in each case. If acquiring in frame $t+1$ yields the highest benefit, then we acquire frame $t+1$ immediately. However, if waiting until frame $t+2$ yields a higher benefit, then we do not acquire frame $t+1$, and instead repeat the above procedure to determine whether we should acquire frame $t+2$ or $t+3$.

Algorithm 3 *Input*: t, the frame number of the last acquired frame. *Output*: $t+k$, the frame number of the next frame to acquire

```
$k = 0$
**repeat**
    $k = k + 1$
    set $B_{now}$ to benefit of acquiring frame $t + k$
    set $B_{later}$ to benefit of acquiring frame $t + k + 1$
**until** $B_{later} < B_{now}$
**return** $t + k$
```

We demonstrate the effectiveness of this algorithm experimentally in Section VI-A2.

F. When to Stop Acquiring

To save time and reduce unnecessary photobleaching, we must know when to stop acquisition. We would want to do this when the model has been learned with high confidence, or when future acquisitions were not expected to give much improvement, particularly if the expected cost of those future acquisitions is high.

1) Single Object Model: For the case of a single object model, we can estimate the reward of future acquisitions (see Section V-C), and so the obvious stopping criterion is to cease acquisition when the expected cost of acquiring the next frame exceeds the expected reward of that frame. In practice this is too simplistic: it is possible that the reward of acquiring in the current frame will be less than the cost, but that the reward of acquiring in future frames will still exceed the cost. To account for this, we only cease acquisition if the expected reward of acquiring in frame $t+k$ is below the expected cost for all k up to some constant. This is expressed formally in Algorithm 4, and in Section VI-A3 we achieve good results experimentally using $k = 5$.

Algorithm 4 *Input*: k, the number of frames to look ahead to. *Output*: c, a boolean indicating whether to continue acquiring

```
**for** $i = 1$ to $k$ **do**
    set $R_i$ to reward of acquiring frame $t + i$
    set $C_i$ to cost of acquiring frame $t + i$
**end for**
$c = \max_i[(R_i/C_i)] > 1$
**return** $c$
```

2) Multiple Object Model: With multiple object models, we do not have an easy way to predict the reward of a future acquisition. Instead, we look at how much recent acquisitions have improved the model estimate. Specifically, we try to determine whether our model estimate at frame $t-1$ is significantly different (or better) than our model estimate at frame $t-2$. One way to measure this is to take the distance between the two models using (7), which gives an indication of change (albeit not of improvement). However, when testing on real data in Section VI-B3, we found an even better measure was to compare the likelihood of the observed data in frame t given the model from frame $t-1$, and given the model from frame $t-2$. If the former is not greater than the latter by a given threshold, for k successive frames, we stop acquiring. Experimentally, we found $k = 3$ was sufficient. Note that this method implicitly assumes there are a large number of objects present (at least 30), because otherwise the likelihood estimates would be too noisy.

VI. Experimental Results

We now present the results of experiments to test our proposed framework and algorithms. We divide this section into experiments on single object models (which use synthetic data), and experiments on multiple object models (which use real data).

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

A. *Single Object Model*

The experiments in this section are based on synthetic data. As our goal in this paper is to set up the framework and determine its potential objectively, we need accurate ground truth. This is only possible with synthetic data. Experiments on real data appear in Section VI-B.

All synthetic data is in 3-D and with frames of size $1000 \times 1000 \times 15$. When generating this data, we assume that an object is equally likely to move under an RW model or a CV model. The range of velocities used varies from 0 to 2 pixels/frame in each dimension. The ranges of (x, y, z) variances used are from 0 to 10, and the diagonal covariance matrix created from these values is then randomly rotated in (x, y, z) to create a general covariance matrix. For each experiment, we generate a new set of synthetic data.

1) Where to Acquire: We now test Algorithm 2, which determines where to acquire. Our goal here is to reduce photobleaching, and so we define the costs such that exposing an object (photobleaching cost) carries $10\times$ the cost of acquiring a frame (time cost). We compare Algorithm 2 to three alternative acquisition algorithms: 1) acquire all pixels, 2) acquire k pixels in each frame, choosing those most likely to contain the object, 3) acquire k pixels in each frame, choosing them randomly. When we ran our intelligent algorithm on a synthetic data set, it ended up acquiring about 25% of the pixels, and so we choose k to be 25% of the total number of pixels. All algorithms use the same frame rate.

To run this experiment, we first simulate an object as described at the beginning of Section VI-A, and then attempt to learn the parameters $(\mathbf{v}, \mathbf{\Sigma})$. Fig. 4(a) shows the error of our prediction with respect to the frame number when using our intelligent algorithm, and when using our three comparison algorithms. These results are averaged over 50 trials, each of a different simulated track. As expected, acquiring all pixels results in the lowest error after 100 frames, with our intelligent algorithm performing second. Choosing the pixels randomly performs worst. However, the goal here is not to speed up model learning, but rather to reduce photobleaching. In Fig. 4(b), we see these same results plotted against the photobleaching incurred during acquisition. Because there are 15 z-slices, a standard acquisition will expose the object 15 times. For convenience, we set the photobleaching cost per exposure, ρ, to be 1/15, so that acquiring all the pixels in a frame will contribute a total photobleaching cost of 1. As a result, after 100 frames the photobleaching cost of the dotted black line reaches 100, but the other curves stop short of 100 because they do not acquire every pixel, and, thus, their maximum photobleaching cost is less. For any given photobleaching cost, our intelligent algorithm achieves a lower model error than any of the comparison algorithms.

The computation time for the intelligent algorithm with 100,000 particles is about 0.13 s per frame.

Note that if the time cost was large and the photobleaching cost small, the algorithm would acquire every pixel in the frame, and, thus, we would get the curves represented by the dotted black line. Therefore, as the time cost increases, the curve will shift from that of the solid red line towards that of the dotted black line.

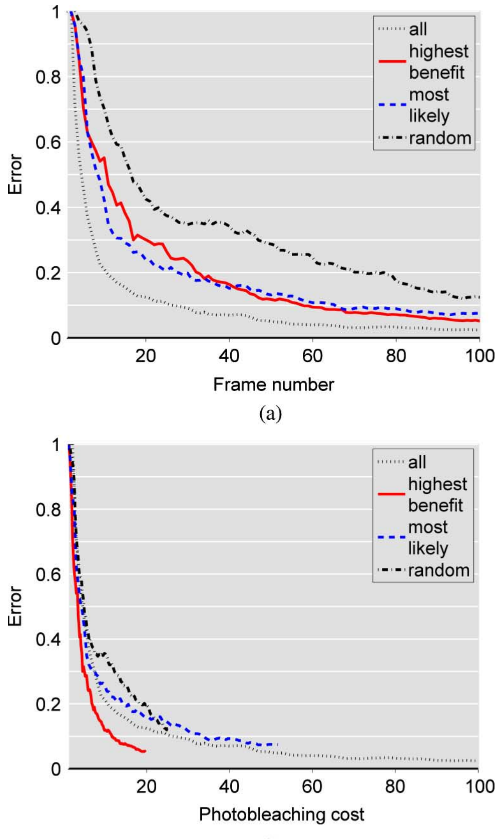


Fig. 4. Where to acquire (single object, synthetic data). These curves show the average rate at which a model is learned, each using a different acquisition strategy. There were 50 trials. Plot (a) shows error against frame rate. Plot (b) shows error against photobleaching incurred. The dotted black line shows the method where every pixel in every frame was acquired. The solid red line shows the method from Algorithm 2. The dashed blue line shows when the 25% of the pixels most likely to contain the object are acquired. The dashdotted black line shows when 25% of pixels are acquired randomly. We can see that, for a given amount of photobleaching, Algorithm 2 gives the lowest error.

2) When to Acquire: Our next experiment tests Algorithm 3. Once again, our goal is to reduce photobleaching, and so we define the costs such that exposing an object (photobleaching cost) carries $10\times$ the cost of acquiring a frame (time cost). We use this algorithm to build a model for a single object moving under the conditions described at the beginning of Section VI-A. We repeat this in 50 trials, each with a different simulated object. We also build a model for the same 50 simulated objects using an algorithm that simply acquires at a constant frame rate. This constant frame rate is chosen so that both algorithms acquire the same number of frames overall, which means that they each incur the same amount of photobleaching overall.

We can see in Fig. 5(a) that Algorithm 3 (solid red line) outperforms the constant frame rate algorithm (dashed blue line), even though they each acquire the same number of frames. As a comparison, we have also shown the rate of learning when every frame is acquired (dotted black line). Although this is also a constant frame rate algorithm, it acquires more overall frames than the other two algorithms. Plot (b) shows these same results

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

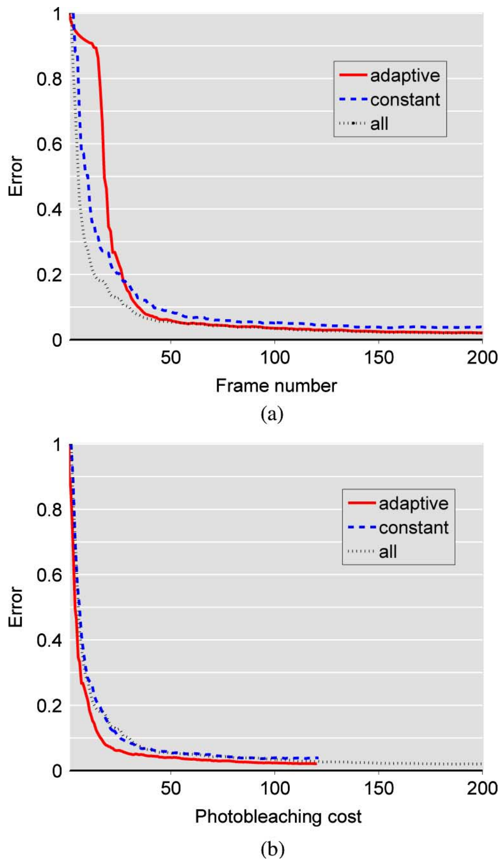


Fig. 5. When to acquire (single object, synthetic data). The solid red line shows the average error when we intelligently choose when to acquire. The dashed blue line shows the average error when we acquire at a constant frame rate that acquires the same overall number of frames as the intelligent algorithm. The dotted black line shows the error when we acquire every frame. Results are averaged over 50 trials. Plot (a) shows error against frame rate. Plot (b) shows error against photobleaching incurred. We can see that, for a given amount of photobleaching, the intelligent algorithm gives the lowest error.

plotted against photobleaching. For any given photobleaching cost, Algorithm 3 achieves a lowest error.

With 100,000 particles, the computation time for the intelligent algorithm is about 0.12 s per frame. On average (over the 50 trials), the algorithm acquired 120 of the 200 frames. Of these frames, 61% of the time a frame was acquired immediately after the previous frame, 26% of the time there was a gap of one frame, and 12% of the time there was a gap of two frames. The remaining 1% is when three or more frames are skipped in succession, which occurs mainly at the beginning of acquisition.

3) When to Stop Acquiring: We now test Algorithm 4, which determines when to stop acquiring. Here, we simulate 100 time series under the conditions described at the beginning of Section VI-A. We compare two algorithms for learning this model: the control algorithm stops acquiring after a fixed number of frames; our intelligent algorithm stops acquiring only when the expected cost exceeds the expected reward for the subsequent five frames. Both algorithms acquire at the same frame rate. The results are shown in Fig. 6. Over the 100 time series this figure shows the average number of frames acquired by each algorithm, and the average error at the time that acquisition stopped. Note that, in the intelligent case, the algorithm has acquired a different number of frames for each time series, and it is the

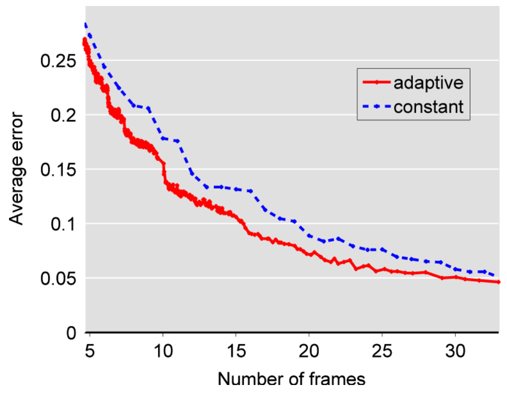


Fig. 6. When to stop acquiring (single object, synthetic data). The dashed blue line shows the average error when we acquire for a given number of frames. The solid red line shows the average error for a given average number of frames when we intelligently choose when to stop acquiring. The 500 data points on this curve are obtained by varying the cost of acquiring each frame, τ, from 0.02 (leftmost point) down to 0.0005 (rightmost point). This intelligent algorithm results in a lower average error for any given average number of frames.

average number of frames acquired that is being plotted on the x-axis. In the case of the intelligent algorithm, there are 500 different points on the curve, which are obtained by varying the cost of acquiring each frame, τ, from 0.02 down to 0.005. In the case of the control algorithm, the different points are obtained by varying the number of frames that are acquired each time. We can see that the intelligent algorithm (solid red line) consistently gives better results than the control algorithm (dashed blue line).

With 100,000 particles, the computation time of this algorithm is about 0.18 s per frame.

B. Multiple Object Model

We now test our algorithms for multiple object models. Although these algorithms do not have such a strong theoretical foundation as the single object models, all experiments in this section are on real data. This raises the issue that we do not have the ground truth, and, hence, we make the assumption that the true model is the model that we get after acquiring all available frames in the time series.

We have 6 time series available. Each are of vesicles moving in 3T3 mouse cells. There are 15 z-slices in each time series, at a resolution of 0.5 microns. Each of these slices is of size 1024×1080, with a pixel resolution of 0.11 microns. The number of objects present in the time series range from 300 to 1000 (with variance of about 10% within a time series). The number of time points (frames) available in each time series ranges from 20–23. Except for those experiments that use all 6 time series, results in this section are always presented for the time series with 23 frames.

1) Model Verification: In Section IV-C, we outlined a method to test the effectiveness of our models. Specifically, we measured the likelihood of the observed data in each frame for a given model. Fig. 7 compares these likelihoods for five different models. The dotted black line represents a constant model, which assumes that objects move up to 32 pixels in each of the x,y-dimensions, and up to 2 pixels in the z-dimension, all with equal probability. The dashed blue line assumes that an

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

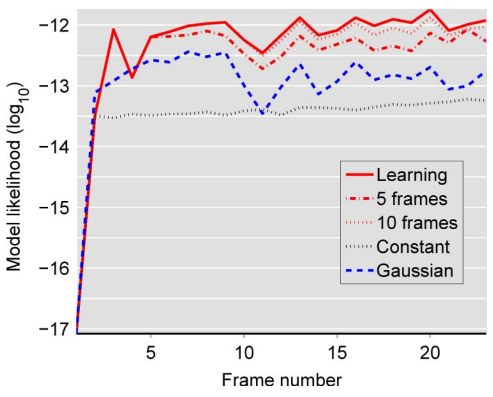


Fig. 7. Model verification (multiple objects, real data). This plot shows the log likelihood of the observed data at any given frame, normalized by the number of objects. The dotted black line is for the model that assumes objects have an equal probability of moving anywhere within a $63 \times 63 \times 5$ window. The dashed blue line is for the model that assumes the probability is governed by a Gaussian with an xy-standard deviation of 2 pixels and a z-standard deviation of 0.4 pixels. The solid red line learns the model on-the-fly using Algorithm 1. The dashdotted red line uses this same algorithm but stops learning after 5 frames. The dotted red line stops learning after 10 frames. We can see that the models from this algorithm give the highest likelihoods, with continued learning improving them further.

object's motion will be Gaussian, with a standard deviation of 2 pixels in the x,y-dimension, and 0.4 pixels in the z-dimension. This size was chosen because, under the restriction that the x,y standard deviations are equal, it provides the best average fit over all 6 time series. This model outperforms the constant model. The solid red line represents the model learned using our model building algorithm from Section IV-C (Algorithm 1). The model at a given frame is built using all the observed data up to (but not including) the current frame. We can see that the model output by this algorithm outperforms both the constant model and the Gaussian model. The dashdotted red line and dotted red line show the results of the same algorithm but when learning ceases after 5 and 10 frames, respectively. Because these lines are below the solid red line, we can see that continued learning does still provide benefit, and this holds true right up to the last frame in the time series.

The final model output using Algorithm 1 for this time series is as follows:

$$\mathbf{v} = [0.17 \quad -0.07 \quad -0.09]$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} 6.39 & 0.26 & -0.03 \\ 0.26 & 2.96 & -0.02 \\ -0.03 & -0.02 & 0.46 \end{bmatrix}.$$

2) Where to Acquire: We now test the method of Section V-D2, which aims to reduce the photobleaching incurred while learning a model, as well as to acquire frames at a faster rate. Here, instead of acquiring every pixel in a frame, we skip pixels with a low probability of containing an object, and assume that they do not contain objects. Hence, our acquisition set consists of those pixels most likely to contain an object, and we make the set large enough such that we expect to capture $k\%$ of the objects. As values of k, we choose 100,98,85,75. Note that when $k = 100$, all pixels are acquired, because this is the only way to guarantee capturing all the objects.

Although we test this method on real data, we have not yet implemented the intelligent acquisition protocol on an actual microscope. Instead, the microscope acquires all pixels in a frame, but then we simulate intelligent acquisition by only storing the given subset of these pixels. Note that this results in slightly different pixel values than if we had implemented the protocol on the microscope, because the timing of each pixel acquisition is altered, and the incurred photobleaching is altered. Furthermore, because frames are acquired faster when we only acquire a subset of pixels, we could potentially increase the frame rate on an actual microscope when using this algorithm.

As for the simulated data, we define photobleaching as being proportional to the number of times the object is exposed. We then estimate the number of object exposures that would have occurred had we acquired intelligently.

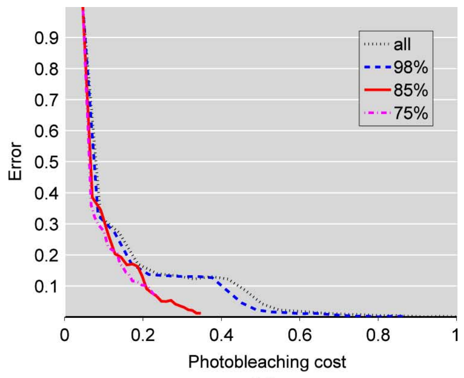
As mentioned in Section IV-B1 , the model can be built in about 0.05 s. Using this model to predict the probability of any pixel containing an object, and then choosing the appropriate pixels to acquire, takes about 0.5 s.

In Fig. 8(a), we see a plot of the error against time for the 23 frames of the time series. We see that choosing $k = 98$ causes the model to be learned at the same rate as $k = 100$ (these plots are so close that the reader may not be able to distinguish them). For $k = 85$ and $k = 75$, we can see that the rate of learning is slower. In plot (b), we see a plot of this same error against photobleaching. Here, we see that $k < 100$ gives a lower error for the same amount of photobleaching, with the optimal value being $k = 85$. In fact, $k = 85$ was the optimal value for all k tested, and a similar value held across all 6 time series. As k gets lower, we expect that the model error will never reach 0 because the assumption that unacquired pixels do not contain objects becomes increasingly invalid. Although we can demonstrate this on synthetic data, we do not have enough time points to show it on real data.

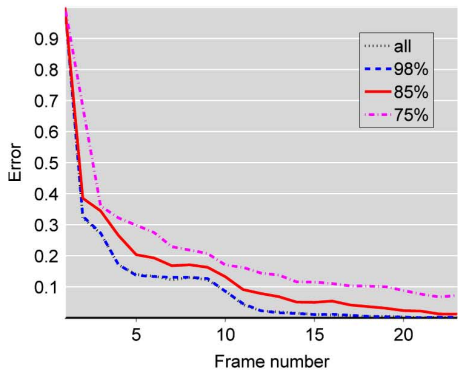
As well as reducing photobleaching, the reduced number of pixels acquired should allow for faster frame acqusition. The total percentage of pixels acquired in this experiment for $k = \{100,98,85,75\}$ was $\{100\%,7.6\%,2.5\%,1.7\%\}$, respectively. This should allow for increased temporal resolution, which, in turn, would make it easier to predict where objects will be (because they will not have moved as far), thus allowing us further reduce the number of pixels acquired, and so forth. Note that the sharp drop from 100% to 7.6% reflects that the majority of pixels in a frame do not contain objects, and that many of these pixels can be reliably predicted.

3) When to Stop Acquiring: Finally, we test our algorithm to determine when to stop acquiring. Table I shows one set of results. For this set of results, we stopped acquiring a time series when, for three successive frames, the likelihood of the observed data given the current model did not exceed the likelihood given the previous model by more than 1.02^n, where n is the number of objects in the frame. In the left side of Table I, we show the number of frames acquired for each of the 6 time series, and the final error in each case. We see that on average, 12.83 frames were acquired. On the right side, we acquire exactly 13 frames for each time series (rounding up from 12.83). We can see that the average final error is higher in this case than when we used

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.



(a)



(b)

Fig. 8. Where to acquire (multiple objects, real data). These curves show the rate at which a model is learned. In each case, we acquire the pixels with the highest likelihood of containing an object. The dashdotted magenta line acquires just enough pixels so that we expect 75% of the objects to be captured. The solid red line acquires enough so that we expect 85% of the objects to be captured. The dashed blue line aims for 98%, and the dotted black line acquires all pixels (and, thus, captures all objects). Plot (a) shows error against frame rate. Plot (b) shows error against photobleaching incurred. We can see that the solid red line converges to almost zero error, and does it with less photobleaching incurred.

our adaptive algorithm, even though the same total number of frames have been acquired.

Although Table I used 1.02^n as a threshold, we can also use different bases for our threshold. We repeated the experiment for thresholds $\{1.2^n, 1.1^n, 1.05^n, 1.04^n, 1.02^n, 1.01^n, 1.008^n, 1.006^n, 1.004^n\}$, and plotted the results in Fig. 9. In all cases, we see that the adaptive algorithm gives lower final errors than the constant algorithm.

The only extra computation required is to store the model from the previous frame and evaluate the likelihood given this model. This consists of a simple matrix multiplication and, thus, takes negligible time.

VII. Conclusion

We presented a framework and a set of algorithms to learn and intelligently acquire models of fluorescence microscope data, a problem not addressed until now. As large amounts of 2-D and 3-D times series are being acquired every day, reducing acquisition time and total light exposure is a desirable goal. Increased light exposure to objects leads to an increase in photobleaching and phototoxicity, which limit the duration over which we can acquire. Moreover, our goal during acquisition is to learn *models* rather than reconstruct *images*. We use an active learning approach to determine *where* and *when* to acquire, as well as when to *stop* acquiring. We test our framework both on synthetic data, as the only source of ground truth on which we can objectively assess the performance of our algorithms, as well as on real data to demonstrate its real-world practicability. Results show great promise both in terms of accuracy of the models being acquired, as well as in terms of efficiency of acquisition. Future work will concentrate on extending the class of models learned, and improving the rigorousness of the algorithms for multiple object scenarios.

TABLE I
When to Stop Acquiring (Multiple Objects, Real Data).

	Adaptive		Constant	
Cell	Frames	Error	Frames	Error
1	15	0.0002	13	0.0041
2	9	0.0039	13	0.0000
3	8	0.0383	13	0.0103
4	15	0.0114	13	0.0231
5	20	0.0002	13	0.0443
6	10	0.0504	13	0.0272
Avg	12.83	0.0174	13	0.0218

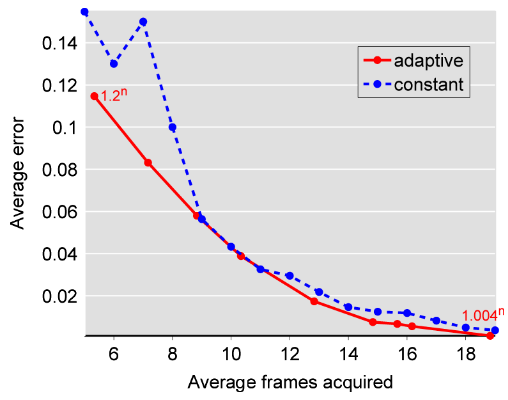


Fig. 9. When to stop acquiring (multiple objects, real data). The dashed blue line shows the average error when we acquire for a given number of frames. The solid red line shows the average error for a given average number of frames when we intelligently choose when to stop acquiring, with stopping thresholds $\{1.2^n, 1.1^n, 1.05^n, 1.04^n, 1.02^n, 1.01^n, 1.008^n, 1.006^n, 1.004^n\}$. This intelligent algorithm results in a lower average error for any given average number of frames.

VIII. Reproducible Research

To facilitate sharing the method with end users as well as developers, we provide the code and information necessary to reproduce the results in this paper at [25].

Acknowledgment

The authors would like to thank E. Glory, F. Lanni, and G. Rohde, for helpful insights and discussion. They would also like to thank the anonymous reviewers for their constructive comments and valuable feedback that have had a significant effect on this work.

References

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

[2] T. Zhao and R. Murphy, "Automated learning of generative models for subcellular location: Building blocks for systems biology," *Cytometry*, vol. 71A, pp. 978–990 [Online]. Available: http://www3.interscience.wiley.com/cgi-bin/abstract/116835310/ABSTRACT

[3] R. Goldman and D. Spector, *Live Cell Imaging: A Laboratory Manual*. Cold Spring Harbor, NY: CSHL Press, 2004.

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

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.

[5] M. V. Boland, M. Markey, and R. F. Murphy, "Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images," *Cytometry*, vol. 33, pp. 366–375 [Online]. Available: http://murphylab.web.cmu.edu/publications/69-boland1998.pdf

[6] M. V. Boland and R. F. Murphy, "A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells," *Bioinformatic*, vol. 17, no. 12, pp. 1213–1223, 2001.

[7] E. Glory and R. F. Murphy, "Automated subcellular location determination and high throughput microscopy," *Developmental Cell* , vol. 12, pp. 7–16 [Online]. Available: http://www.cell.com/developmental-cell/abstract/S1534-5807(06)00570-3

[8] A. Chebira, Y. Barbotin, C. Jackson, T. E. Merryman, G. Srinivasa, R. F. Murphy, and J. Kovačević, "A multiresolution approach to automated classification of protein subcellular location images," *BMC Bioinformatics*, vol. 8, no. 210, 2007.

[9] R. A. Hoebe, C. H. V. Oven, T. W. J. Gadella, Jr, P. B. Dhonukshe, C. J. V. Noorden, and E. M. Manders, "Controlled light-exposure microscopy reduces photobleaching and phototoxicity in fluorescence live-cell imaging," *Nature Biotech.*, vol. 25, no. 2, pp. 249–53, 2007.

[10] Z.-P. Liang and P. C. Lauterbur, "An efficient method for dynamic magnetic resonance imaging," *IEEE Trans. Med. Imag.*, vol. 13, pp. 677–686, 1994.

[11] L. P. Panych, C. Oesterle, G. P. Zientara, and J. Hennig, "Implementation of a fast gradient-echo SVD encoding technique for dynamic imaging," *Magn. Res. Imag.*, vol. 35, no. 4, pp. 554–62, 1996.

[12] J. Tsao, P. Boesiger, and K. P. Pruessmann, "Kt BLAST and kt SENSE: Dynamic MRI with high frame rate exploiting spatiotemporal correlations," *Magn. Res. Imag.*, vol. 50, no. 5, pp. 1031–1042, 2003.

[13] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.

[14] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proc. ACM SIGIR Conf. Research and Development in Information Retrieval*, 1994, pp. 3–12.

[15] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. Workshop Comp. Learn. Theory*, 1992, pp. 287–294.

[16] B. Anderson and A. Moore, "Active learning for hidden Markov models: Objective functions and algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 9–16.

[17] M. West and J. Harrison*, Bayesian Forecasting and Dynamic Models*. New York: Springer, 1997.

[18] C. Berzuini, N. Best, W. Gilks, and C. Larizza, "Dynamical conditional independence models and Markov chain Monte Carlo methods," *J. Amer. Statist. Assoc.*, vol. 92, no. 440, 1997.

[19] C. M. Bishop*, Pattern Recognition and Machine Learning, ser. Information Science and Statistics*. New York: Springer, 2006.

[20] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *Proc. IEEE Radar and Signal Processing*, 1993, vol. 140, no. 2, pp. 107–113.

[21] A. Doucet, On Sequential Simulation-Based Methods for Bayesian Filtering Cambridge Univ, Cambridge, U.K., 1998.

[22] P. Muller, "Monte Carlo integration in general dynamic models," *Contemp. Math.*, vol. 115, pp. 145–163, 1991.

[23] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Amer. Statist. Assoc.*, vol. 94, no. 446, pp. 590–591, 1999.

[24] M. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S. Adelaide, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[25] C. Jackson, R. F. Murphy, and J. Kovačević, Intelligent Acquisition and Learning of Fluorescence Microscope Data Models [Online]. Available: http://www.andrew.cmu.edu/user/jelenak/Repository/08_JacksonMK/08_JacksonMK.html

Charles Jackson (S'06) received the B.S. degree in electrical and computer engineering from the University of Canterbury, Christchurch, New Zealand, in 2003. He is currently pursuing the Ph.D. degree in biomedical engineering at Carnegie Mellon University, Pittsburgh, PA.

His research interests include the automated analysis and intelligent acquisition of fluorescence microscope images.

Robert F. Murphy (M'02–SM'07) is the Ray and Stephanie Lane Professor of Computational Biology and Director of the Ray and Stephanie Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA. He also is a Professor of biological sciences, biomedical engineering, and machine learning and directs (with Ivet Bahar) the joint CMU-Pitt Ph.D. Program in Computational Biology. He has co-edited two books and published over 150 research papers. His career has centered on combining fluorescence-based cell measurement methods with quantitative and computational methods. His group at Carnegie Mellon pioneered the application of machine learning methods to high-resolution fluorescence microscope images depicting subcellular location patterns in the mid 1990s. This work led to the development of the first systems for automatically recognizing all major organelle patterns in 2-D and 3-D images. He currently leads NIH-funded projects for proteome-wide determination of subcellular location in 3T3 cells (with P. Berget and J. Jarvik) and continued development of the SLIF system for automated extraction of information from text and images in online journal articles (with W. Cohen and E. Xing).

Dr. Murphy is a Fellow of the American Institute for Medical and Biological Engineering and an Alexander von Humboldt Foundation Research Award honoree. He is President of the International Society for Advancement of Cytometry and was named as the first External Senior Fellow of the School of Life Sciences in the Freiburg (Germany) Institute for Advanced Studies in 2008.

Jelena Kovačević (S'88–M'91–SM'96–F'02) received the Dipl. Electr. Eng. degree from the EE Department, University of Belgrade, Yugoslavia, in 1986, and the M.S. and Ph.D. degrees from Columbia University, New York, in 1988 and 1991, respectively.

She is a Professor of biomedical engineering and electrical and computer engineering and the Director of the Center for Bioimage Informatics, Carnegie Mellon University, Pittsburgh, PA. Her research interests include bioimaging as well as multiresolution techniques such as wavelets and frames. From 1991–2002, she was with Bell Labs, Murray Hill, NJ. She was a co-founder and Technical VP of xWaveforms, based in New York City. She was also an Adjunct Professor at Columbia University. In 2003, she joined Carnegie Mellon University. She is a coauthor (with M. Vetterli) of the book *Wavelets and Subband Coding* (Prentice Hall, 1995).

Dr. Kovačević coauthored a top-10 cited paper in the *Journal of Applied and Computational Harmonic Analysis*, and the paper for which A. Mojsilovic received the Young Author Best Paper Award. Her paper on multidimensional filter banks and wavelets (with Martin Vetterli) was selected as one of the Fundamental Papers in Wavelet Theory. She received the Belgrade October Prize in 1986 and the E.I. Jury Award at Columbia University in 1991. She was the Editor-in-Chief of the IEEE TRANSACTIONS ON IMAGE PROCESSING. She served as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, as a Guest Co-Editor (with I. Daubechies) of the Special Issue on Wavelets of the PROCEEDINGS OF THE IEEE, Guest Co-Editor (with M. Vetterli) of the Special Issue on Transform Coding of the *IEEE Signal Processing Magazine*, and Guest Co-Editor (with R. F. Murphy) of the Special Issue on Molecular and Cellular Bioimaging of the *IEEE Signal Processing Magazine*. She is/was on the Editorial Boards of the *Foundations and Trends in Signal Processing*, SIAM book series on Computational Science and Engineering, *Journal of Applied and Computational Harmonic Analysis*, *Journal of Fourier Analysis and Applications*, and the *IEEE Signal Processing Magazine*. She is a regular member of the NIH Microscopic Imaging Study Section. From 2000–2002, she served as a Member-at-Large of the IEEE Signal Processing Society Board of Governors. She is the Chair of the Bio Imaging and Signal Processing Technical Committee. She was the General Chair of ISBI 06, General Co-Chair (with V. Goyal) of the DIMACS Workshop on Source Coding and Harmonic Analysis and General Co-Chair (with J. Allebach) of the Ninth IMDSP Workshop. She is/was a plenary/keynote speaker at the "20 Years of Wavelets" 09, European Women in Mathematics 09, MIAAB Workshop 07, Statistical Signal Processing Workshop 07, Wavelet Workshop 06, NORSIG 06, ICIAR 05, Fields Workshop 05, DCC 98, as well as SPIE 98.

Authorized licensed use limited to: Carnegie Mellon Libraries. Downloaded on September 10, 2009 at 19:29 from IEEE Xplore. Restrictions apply.