The New England Journal of Statistics in Data Science Volume 4, 204–219 (2026)
DOI: https://doi.org/10.51387/26-NEJSDS97

Unsupervised Cell Segmentation by Fast Gaussian Processes

Laura BARACALDO*, Blythe KING*, Haoran YAN*, Yizi LIN, Nina MIOLANE, and Mengyang GU**

Abstract

Cell boundary information is crucial for analyzing cell behaviors from time-lapse microscopy videos. Existing supervised cell segmentation tools, such as ImageJ, require tuning various parameters and rely on restrictive assumptions about the shape of the objects. While recent supervised segmentation tools based on convolutional neural networks enhance accuracy, they depend on high-quality labeled images, making them unsuitable for segmenting new types of objects not in the database. We developed a novel unsupervised cell segmentation algorithm based on fast Gaussian processes for noisy microscopy images without the need for parameter tuning or restrictive assumptions about the shape of the object. We derived robust thresholding criteria adaptive for heterogeneous images containing distinct brightness at different parts to separate objects from the background, and employed watershed segmentation to distinguish touching cell objects. Both simulated studies and real-data analysis of large microscopy images demonstrate the scalability and accuracy of our approach compared with the alternatives.

KEYWORDS AND PHRASES: Bayesian inference, Image segmentation, Lattice, Microscopy, Scalability.

1. INTRODUCTION

The spatial organization of cells, such as orientation, density, and shape, are fundamental to characterizing physical mechanisms and biological functions [8, 37], as the arrangement of cells influences their functions, including proliferation and differentiation [30]. For example, high cell density can signal cells to die to maintain tissue homeostasis [9], and cell shape has been linked to DNA synthesis and cell growth [12]. Advances in microscopy techniques enable detailed video recordings of cellular motions. Image segmentation approaches have been developed for microscopy images [33], where the segmentation results, including cell locomotion, alignment, and density, can be used for modeling cell behaviors [47, 15, 10, 25]. Therefore, precise cell image segmentation is essential for quantifying cell location, alignment, and morphology, as these statistics enhance our understanding of how cellular spatial organization influences human health outcomes [53, 2].

Cellular segmentation tools can be broadly classified as unsupervised and supervised methods, based on whether labeled data is needed for training the models. One of the most popular tools of unsupervised cellular segmentation is the ImageJ [1, 46, 44], which enables segmenting both cell nuclei and whole cells, and linking cells across frames of a microscopy video via plug-ins such as TrackMate [51]. Prior to using ImageJ for cell detection and segmentation, a series of user-driven image pre-processing steps must be completed in the program. For instance, the image is typically converted into a binary image first, based on the threshold chosen either manually or automatically via several global threshold setting methods [34, 29]. Additional image processing steps, such as despeckling for noise reduction [11], typically need to be run before segmentation. To detect cell objects within an image, the algorithm scans the corresponding binary image until the edge of an object is reached. Then, the algorithm traces around the edge of the object until the starting point is reached. This process is repeated for pixels of each object, or the foreground pixels, in the binary image. The centroid of each cell object can be identified by fitting an ellipse for each object using the second-order central moments [46], which assumes the cell boundary is elliptical. The parameters for the best-fit ellipse, including the major and minor axes and centroids, and matrices for the object masks for each object boundary, are recorded in ImageJ [11]. Additional functionality, such as extension of elliptical outlines, can be added to the user interface by writing macro code to streamline image analysis or adding third-party plug-ins [46]. However, achieving reasonable segmentation results is often labor-intensive and requires human interaction at each step. Furthermore, the assumption of the shape of the cells in ImageJ is restrictive. Though the ellipse can approximate nuclei of certain cells, it cannot approximate the shape of cytoplasm or the whole cell of frequently used cells in experiments, such as fibroblasts or epithelial cells.

Supervised cellular segmentation methods, are typically built upon a convolutional neural network structure, such as the U-net [41]. A popular tool is Cellpose, which builds U-nets for segmenting images that contain one type of cells

*The first three authors contribute equally.
**Corresponding author.

that are trained on a dataset of over 70,000 segmented objects, including annotated images that contain numerous types of cells [50]. The model first compresses the image and then reconstructs the image to its original dimensions through a U-net to capture both local and global features. It further generates a topological map from diffusion simulation to product gradient maps for the model to learn the horizontal and vertical gradients of the image. The final layer of Cellpose produces three feature maps, including the horizontal and vertical directional gradients, and the sigmoid transformation of the probability map of the output image from the U-net, and the parameters of the Cellpose are then estimated by fitting a loss function of the gradient vector and binary label of each pixel of the image. As the Cellpose segmentation requires massive training data, it may not work well for detecting novel cell types, where high-labeled image data that may not be readily available. Furthermore, training Cellpose and using it for segmenting large images with several million pixels can be computationally expensive.

In response to the challenges the current methods face, we present a novel unsupervised cell image segmentation method for segmenting images containing one type of cells, based on separable Gaussian processes, automated thresholding, and watershed operations for image segmentation. The proposed method has several advantages. First, the separable covariance structure on image data enables us to decompose the large covariance into a Kronecker product of two small covariances, which substantially reduces the computational cost for computing the matrix inversion and the logarithm of the determinant of the covariance matrix. Second, unlike conventional unsupervised image segmentation methods such as ImageJ, which require manual tuning of multiple hyperparameters, including Gaussian blur radius, thresholding method, and minimum object size, our proposed segmentation pipeline is free of manual parameter selection. We developed fast algorithms for maximum likelihood estimation of the parameters in GPs and the predictive distribution. The predictive mean surface naturally facilitates foreground-background separation by applying a data-driven threshold with the optimal threshold automatically estimated based on the absolute second difference in foreground pixel counts between the thresholds. Third, the cell masks are segmented using the watershed algorithm, which simulates water flowing through a topographical map of the image to separate cell objects. Fourth, our approach is entirely unsupervised, negating any need for annotated training data, in contrast to the supervised methods, such as Cellpose. Thus, the proposed method is particularly suitable for segmenting new cell types or other biological structures. We studied our methodology using simulated data for noise filtering. We also compared our approaches with the alternatives using real microscopy images for segmenting both the cell nucleus and whole cell in optically dense images, a challenging scenario even when labeled data are available. These studies demonstrate that our approach substantially outperforms the alternatives.

The rest of the article is organized as follows. We discuss GP models of images and a fast algorithm for parameter estimation and predictive distribution in Section 2. In Section 3, we introduce our workflow for cellular image segmentation from Gaussian processes, including image smoothing, binary thresholding, and segmentation. In Section 4, we use simulated data and microscopy images of cells to compare our methods and other alternatives for denoising, including principal component analysis and dynamic mode decomposition. We also demonstrate the computational advantage of the fast algorithm against direct computation. Finally, we generate cell masks for both the cell nucleus and cytoplasm, or the whole cell, by our methods and ImageJ segmentation method to compare their accuracy with the manual labels in Section 5. Our findings not only present a novel cell segmentation technique but also highlight the applicability of fast Gaussian processes in image processing tasks. The data and code of this article are made publicly available: https://github.com/UncertaintyQuantification/cell_segmentation.

2. GAUSSIAN PROCESS MODELS OF IMAGES

2.1 The Likelihood Function and Predictive Distribution

Let us consider a Gaussian process (GP) model for a two-dimensional (2D) microscopy image, where the intensity at the pixel location $\mathbf{x}$ is modeled as $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, with $f(\cdot)$ being a latent Gaussian process (GP) and $\epsilon \sim \mathcal{N}(0, \sigma_0^2)$ being a Gaussian noise with variance σ_0^2 [38]. Consider an $n_1 \times n_2$ image $\mathbf{Y}$, where the (i, j)th term is $y(\mathbf{x}_{i,j})$ for the pixel location $\mathbf{x}_{i,j} = (x_{i,1}, x_{j,2})$. The data matrix can be written as:

$$\mathbf{Y} = \mathbf{F} + \mathbf{E}, \tag{2.1}$$

where $\mathbf{F}$ follows a matrix normal distribution $\mathbf{F} \sim \text{Matrix-Normal}\left(\mu \mathbf{1}_{n_1 \times n_2}, \sigma^2 \mathbf{R}_1, \mathbf{R}_2\right)$ with μ being a mean parameter, $\mathbf{1}_{n_1 \times n_2}$ being a $n_1 \times n_2$ matrix of ones, σ^2 being the variance parameter, $\mathbf{R}_1$ and $\mathbf{R}_2$ being correlation matrices for two inputs, respectively, and the vectorized noise matrix follows $\text{Vec}(\mathbf{E}) \sim \mathcal{MN}(0, \sigma_0^2 \mathbf{I}_{N \times N})$, with $N = n_1 n_2$. Additional trend structure can be modeled in the mean function, and the covariance can be generalized to be semi-separable in the model [14].

Vectorize the observations matrix $\mathbf{y}_v = \text{Vec}(\mathbf{Y})$. After marginalizing out the latent signal matrix $\mathbf{F}$, the observation vector of the image follows a multivariate normal distribution,

$$(\mathbf{y}_v \mid \sigma^2, \eta, \mathbf{R}_1, \mathbf{R}_2) \sim \mathcal{MN}(\mu \mathbf{1}_N, \sigma^2 \tilde{\mathbf{R}}), \tag{2.2}$$

where $\tilde{\mathbf{R}} = \mathbf{R} + \eta \mathbf{I}_N$, $\mathbf{R} = \mathbf{R}_2 \otimes \mathbf{R}_1$, $\otimes$ is a Kronecker product and $\eta = \sigma_0^2 / \sigma^2$ is a nugget parameter. Here the correlation matrices are parameterized kernel functions $R_1(i, i') =$

$K_1(x_{i,1}, x_{i',1})$ and $R_2(j, j') = K_2(x_{j,2}, x_{j',2})$. Frequently used kernel functions include the power exponential kernel and Matérn kernel [38]. Denote $K_l(x_{i,l}, x_{i',l}) = K_l(d_l)$ with $d_l = |x_{i,l} - x_{i',l}|$. For instance, the Matérn kernel with roughness parameter 5/2 follows [20]:

$$K_l(d_l) = \left(1 + \frac{\sqrt{5}d_l}{\gamma_l} + \frac{5d_l^2}{3\gamma_l^2}\right) \exp\left(-\frac{\sqrt{5}d_l}{\gamma_l}\right), \tag{2.3}$$

where γ_l is a range parameter to be estimated for $l = 1, 2$. A GP with the kernel function in Equation (2.3) is twice mean-squared differentiable [38], and it is used as a default choice of some software packages of GP surrogate models [42, 16]. We use the kernel function in (2.3) for demonstration purposes.

After specifying the model, we compute the likelihood function and predictive distribution for parameter estimation and prediction, respectively. The parameters of the model in (2.2) include the mean, variance, range and nugget parameters $\{\mu, \sigma^2, \boldsymbol{\gamma}, \eta\}$ with $\boldsymbol{\gamma} = (\gamma_1, \gamma_2)$ being the range parameters in the kernel function in Equation (2.3). As the number of pixels is large, we employed the maximum likelihood estimator (MLE) to estimate the parameters, as the result is similar to the robust estimation by marginal posterior mode [17]. Given the range and nugget parameters, the MLE of the mean and variance parameters has a closed-form expression: $\hat{\mu} = (\mathbf{1}_N^T \tilde{\mathbf{R}}^{-1} \mathbf{1}_N)^{-1} \mathbf{1}_N^T \tilde{\mathbf{R}}^{-1} \mathbf{y}_v$, where $\hat{\sigma}^2 = \frac{S^2}{N}$ with $S^2 = (\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)^T \tilde{\mathbf{R}}^{-1} (\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)$. The logarithm of the profile likelihood after plugging the MLE of the mean and variance parameters follows:

$$\log(L(\boldsymbol{\gamma}, \eta)) = C - \frac{1}{2}\log(|\tilde{\mathbf{R}}|) - \frac{N}{2}\log(S^2), \tag{2.4}$$

where C is a constant not related to $(\boldsymbol{\gamma}, \eta)$.

The range and nugget parameters are then estimated by maximizing the logarithm of the profile likelihood function:

$$(\hat{\boldsymbol{\gamma}}, \hat{\eta}) = \underset{(\boldsymbol{\gamma}, \eta)}{\operatorname{argmax}} \log(L(\boldsymbol{\gamma}, \eta)). \tag{2.5}$$

Vectorize signal $\mathbf{f}_v = \mathrm{Vec}(\mathbf{F})$ from model (2.1). After plugging the MLE of the parameters, the posterior distribution of the image vector follows

$$(\mathbf{f}_v \mid \mathbf{y}_v, \hat{\mu}, \hat{\sigma}^2, \hat{\eta}, \hat{\boldsymbol{\gamma}}) \sim \mathcal{MN}(\mathbf{f}_v^*, \hat{\sigma}^2 \mathbf{R}^*), \tag{2.6}$$

where the predictive mean and predictive covariance follow

$$\mathbf{f}_v^* = \hat{\mu}\mathbf{1}_N + \mathbf{R}\tilde{\mathbf{R}}^{-1}(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N), \tag{2.7}$$

$$\mathbf{R}^* = \mathbf{R} - \mathbf{R}\tilde{\mathbf{R}}^{-1}\mathbf{R}. \tag{2.8}$$

The predictive mean $\mathbf{f}_v^*$ and predictive variances, the diagonal terms of $\hat{\sigma}^2\mathbf{R}^*$, are required for prediction and uncertainty quantification. However, direct computation of the likelihood function involves inverting the covariance matrix, which takes $\mathcal{O}(N^3)$ operations, where the number of pixels N can be at the order of $10^6 - 10^7$ for microscopy images. A fast computational way without approximation is introduced in Section 2.2 to solve this computational challenge.

2.2 Fast Computation for Images

A wide range of approximation approaches of GPs have been developed [48, 28, 43, 7, 23, 13, 19, 24, 57]. For GPs with product covariances on image data, no approximation is required. Denote the eigendecomposition of the subcovariance $\mathbf{R}_l = \mathbf{U}_l \boldsymbol{\Lambda}_l \mathbf{U}_l^T$, where $\mathbf{U}_l$ and $\boldsymbol{\Lambda}_l$ are matrices of eigenvectors and a diagonal matrix of eigenvalues for $l = 1, 2$, respectively. Furthermore, denote $\boldsymbol{\lambda}_l = (\lambda_{1,l}, \ldots, \lambda_{n_l,l})^T$, an n_l-vector of the diagonal terms in $\boldsymbol{\Lambda}_l$, for $l = 1, 2$.

The log profile likelihood in Equation (2.4) and the predictive distributions in Equation (2.6) can be written as a function in terms of eigenpairs in Lemma 1 below. Lemma 1 eliminates the need to invert the $N \times N$ covariance matrix and reduce the computational complexity to $\mathcal{O}(n_1^3 + n_2^3)$, which is significantly smaller than $\mathcal{O}(n_1^3 n_2^3)$. The proof of Lemma 1 is provided in the Appendix.

Lemma 1. *1. (Profile likelihood). Denote the transformed likelihood vector* $\tilde{\mathbf{y}}_v = \mathrm{Vec}(\tilde{\mathbf{Y}}) = \mathrm{Vec}(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1 \times n_2})\mathbf{U}_2)$. *The logarithm of the profile likelihood in Equation (2.4) can be written as*

$$\log(L(\boldsymbol{\gamma}, \eta)) = C - \frac{1}{2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\log(\lambda_{i,1}\lambda_{j,2} + \eta) - \frac{N}{2}\log\left(\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\frac{\tilde{Y}_{i,j}^2}{\lambda_{i,1}\lambda_{j,2} + \eta}\right), \tag{2.9}$$

where

$$\hat{\mu} = \left(\sum_{j=1}^{n_2}\sum_{i=1}^{n_1}\frac{\tilde{u}_{i,1}^2\tilde{u}_{j,2}^2}{\lambda_{i,1}\lambda_{j,2} + \eta}\right)^{-1} \times \left(\sum_{j=1}^{n_2}\sum_{i=1}^{n_1}\frac{\tilde{u}_{i,1}\tilde{Y}_{i,j,0}\tilde{u}_{j,2}}{\lambda_{i,1}\lambda_{j,2} + \eta}\right), \tag{2.10}$$

with $\tilde{Y}_{i,j}$ *being the* (i,j)*th entry in* $(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1 \times n_2})\mathbf{U}_2)$, $\tilde{Y}_{i,j,0}$ *being the* (i,j)*th entry of the* $n_1 \times n_2$ *matrix* $\mathbf{U}_1^T\mathbf{Y}\mathbf{U}_2$, $\tilde{u}_{i,1}$ *being the* i*th term of the* n_1 *vector* $\mathbf{1}_{n_1}^T\mathbf{U}_1$ $i = 1, \ldots, n_1$ *and* $\tilde{u}_{j,2}$ *being the* j*th term of the* n_2 *vector* $\mathbf{1}_{n_2}^T\mathbf{U}_2$, *for* $j = 1, \ldots, n_2$.

2. (Predictive distribution). The predictive mean vector in Equation (2.7) follows

$$\mathbf{f}_v^* = \hat{\mu}\mathbf{1}_N + \mathrm{Vec}(\mathbf{U}_1\boldsymbol{\Lambda}_1\tilde{\mathbf{Y}}_0\boldsymbol{\Lambda}_2\mathbf{U}_2^T), \tag{2.11}$$

where $\tilde{\mathbf{Y}}_0$ *is a* $n_1 \times n_2$ *with the* (i,j)*th entry being* $\frac{\tilde{Y}_{i,j}}{\lambda_{i,1}\lambda_{j,2}+\eta}$ *for* $i = 1, \ldots, n_1$ *and* $j = 1, \ldots, n_2$. *The pre-*

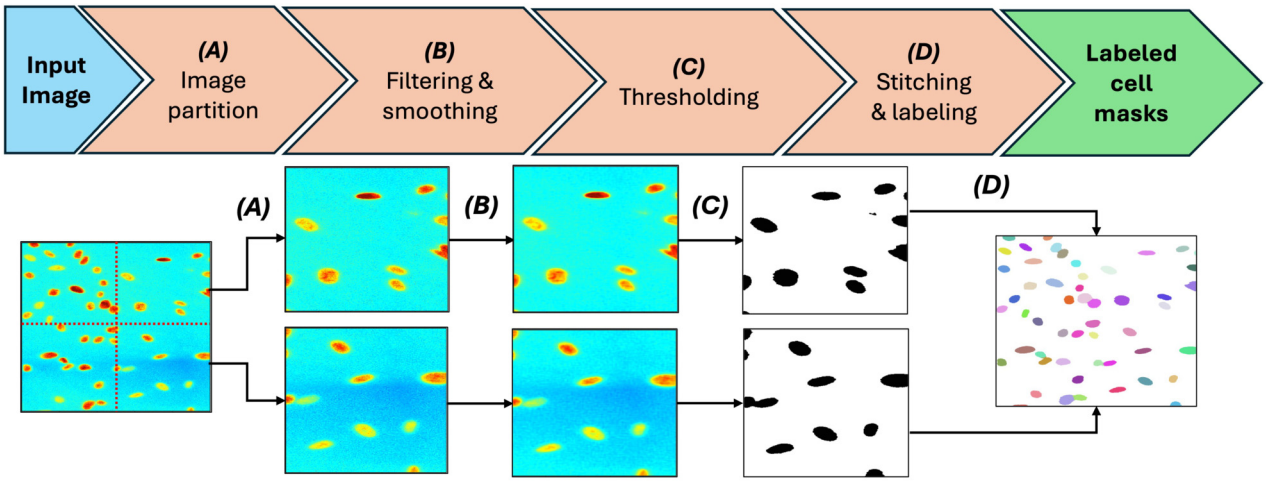


Figure 1: Workflow for segmenting and labeling cell images: (A) Divide the image into different sub-images to enable locally estimated mean and variance parameters for capturing local properties such as the change of brightness. (B) Compute the predictive mean of fast GPs in Section 2.2 to each sub-image, which greatly reduces image noise. (C) Threshold each smoothed sub-image based on the criterion discussed in Section 3.2 to produce binary images, separating cells from the background. The optimal threshold is estimated for each sub-image. (D) Recombine the binary sub-images into a single binary image, and apply the watershed algorithm discussed in Section 3.3 to the image for segmentation and labeling, with each cell marked by a unique color.

dictive variance at the (i,j)*th pixel follows*

$$\hat{\sigma}^2 c^*_{i,j} = \hat{\sigma}^2 \left(1 - \sum_{j'=1}^{n_2} \sum_{i'=1}^{n_1} \frac{\tilde{r}^2_{i,i',1} \tilde{r}^2_{j,j',2}}{\lambda_{i,1}\lambda_{j,2} + \eta}\right)$$

where $\tilde{r}_{i,i',1}$ *is the* i'*th term of the vector* $\mathbf{r}^T_{i,1}\mathbf{U}_1$ *and* $\tilde{r}_{j,j',2}$ *is the* j'*th term of the vector* $\mathbf{r}^T_{j,2}\mathbf{U}_2$*, for* $i' = 1,\ldots,n_1$ *and* $j' = 1,\ldots,n_2$*, with* $\mathbf{r}^T_{i,1} = (K_1(x_{i,1},x_{1,1}),\ldots,K_1(x_{i,1},x_{n_1,1}))^T$ *and* $\mathbf{r}^T_{j,2} = (K_2(x_{j,2},x_{1,2}),\ldots,K_2(x_{j,2},x_{n_2,2}))^T$.

3. IMAGE SEGMENTATION FROM GAUSSIAN PROCESSES

Though Gaussian processes have been widely used in as surrogate models to emulate expensive computer simulations, nonparametric regression, and modeling spatially correlated data [38, 6], the flexibility and scalability of GPs in unsupervised image segmentation have not been studied yet. The overall flow of our cellular image segmentation process is summarized in Figure 1. First, some microscopy images of cells are particularly large with $10^6 - 10^7$ pixels. Similar to other image segmentation methods [46, 50], we segment each large image into multiple sub-images. We then apply the fast algorithm of GPs in Lemma 1 to get the predictive mean of each sub-image, illustrated in Parts (A) and (B) in Figure 1, respectively. We assume the range parameters and nugget of the GPs are the same for each subimage estimated by MLE in Lemma 1, yet the mean and variance parameters are distinct for each separable image, and the estimation of these parameters has a closed-form expression. Such a step enables different estimated parameters to capture the change of optical properties, such as brightness difference and out-of-focus blur from distinct parts of a large microscopy image, which are crucial in practice. Second, we developed a robust measure that will be introduced in Section 3.2 to threshold the predictive mean of each sub-image into background and objects, as shown in Panel (C). Finally, the binary sub-images are restitched in Panel (D), and the watershed algorithm is applied to the binary matrix to retrieve the labeled cell masks, which will be introduced in Section 3.3.

3.1 Image Denoising from Gaussian Processes on Lattice

Let $\mathbf{Y}$ represent the original cell image, which has size $N = n_1 \times n_2$. We denote $y(\mathbf{x}_{i,j})$ as the value of the image at pixel $\mathbf{x}_{i,j} = (x_{i,1}, x_{j,2})$. All possible pixel values are between 0 and 1, as each image in this analysis is in grayscale. Microscopy images of cells can contain more than 10^6 pixels, and some areas of an image can have substantially higher pixel values than other areas. Thus, for a large image $\mathbf{Y}$, it is normally partitioned into a set of cropped images, denoted as $\{\mathbf{Y}_k\}_{k=1}^{\tilde{K}}$, where $\tilde{K}$ is the total number of cropped images as illustrated in Figure 1. This step can substantially reduce the computational cost, and enable segmentation outcomes more adaptive to the features of the local areas in a large image.

As experimental images often contain substantial noise, for each cropped image, we first use GPs with fast computa-

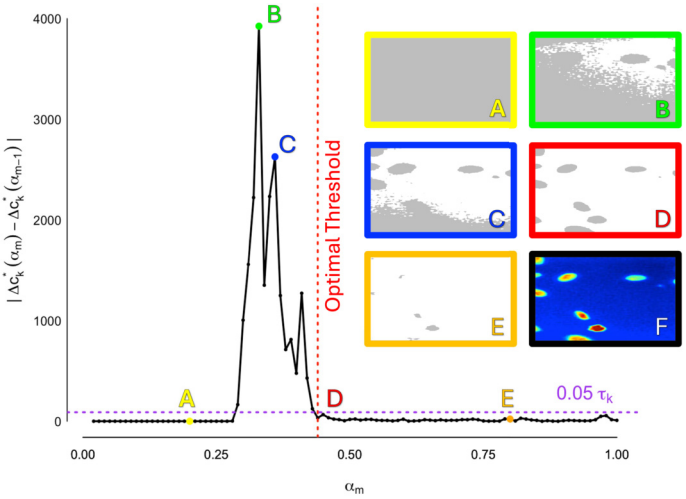


Figure 2: Comparison of binary image results across thresholds based on the absolute second difference in foreground pixels. The threshold, which ranges from 0-1, refers to the proportion of the maximum value of the predictive mean that is set as the binary cutoff. Images A, B, C, D, and E are the binary images generated by setting the corresponding threshold on cropped predictive mean image F. Note that the Image B corresponds with $\text{argmax}_m\, \Delta c_k(\alpha_m)$ in Equation (3.1).

tion introduced in Section 2.1 for denoising the images. The range and nugget parameters of GPs are estimated by maximizing the likelihood in Equation (2.9), and they are held the same for all sub-images as they have similar smoothness properties. The mean and variance parameters are estimated differently for each cropped image to capture the local change of the optical properties. As the sub-images will be turned into a binary image, the discontinuity of the boundary between different sub-images does not impact the image segmentation task.

3.2 Generating Binary Cell Masks

The predictive mean at the interior pixel of a cell object typically has a larger value than the one in the background of a microscopy image. Inspired by the IsoData algorithm [40] in ImageJ [1], we developed an automated and robust way to determine the threshold value for separating objects from background noise using the predictive mean from GP models.

Suppose each cropped image is $n_{k,1} \times n_{k,2}$ with predictive mean $\mathbf{F}_k^*$ with the (i,j)th entry being $F_k^*(\mathbf{x}_{i,j})$ and $\text{Vec}(\mathbf{F}_k^*) = \mathbf{f}_{k,v}^*$ given in Equation (2.11). As the intensity values of the pixels are between 0 to 1 and the predictive uncertainty is small, the range of $F_k^*(\mathbf{x}_{i,j})$ is also approximately between 0 to 1. As the image has only one type of cells, the change of pixel values $F_k^*(\mathbf{x}_{i,j})$ is large if $\mathbf{x}_{i,j}$ is the pixel at the boundary of a cell object, and small elsewhere. Thus we first compute the normalized predictive mean exceeds each threshold value:

$$c_k(\alpha_m) = \sum_{i,j} 1_{\frac{F_k^*(\mathbf{x}_{i,j})}{\max(\mathbf{F}_k^*)} > \alpha_m},$$

where $\{\alpha_m\}_{m=1}^M$ a sequence of equally spaced thresholds from 0 to 1 with the default value $M = 100$, and $\max(\mathbf{F}_k^*)$ denotes the maximum value of the predictive mean $\mathbf{F}_k^*$ of the kth cropped image for $k = 1, \ldots, \tilde{K}$. In general, the range of $\{\alpha_m\}_{m=1}^M$ can be chosen to be the range of $F_k^*(\mathbf{x}_{i,j})/\max(\mathbf{F}_k^*)$. Subsequently, we calculate the difference $\Delta c_k(\alpha_m)$ in pixel counts below

$$\Delta c_k(\alpha_m) = c_k(\alpha_{m-1}) - c_k(\alpha_m).$$

As $\Delta c_k(\alpha_m)$ may not be smooth over α_m, we compute the predictive mean of $\Delta c_k(\alpha_m)$, denoted by $\Delta c_k^*(\alpha_m)$, via a GP model with the default Matérn kernel in Equation (2.3) by the RobustGaSP package [16].

We define α^* as the smallest threshold α_m after the point of maximum fluctuation, denoted as $\arg\max_m \Delta c_k^*(\alpha_m)$, where the smoothed differences $\Delta c_k^*(\alpha_m)$ stabilize within a specified tolerance relative to the variability of $\Delta c_k(\alpha_m)$, an example of which is plotted in Figure 2. Formally, let τ_k represent the standard deviation of the differences $\Delta c_k^*(\alpha_m)$. The optimal threshold α_k^* is given by:

$$\alpha_k^* = \min_m \{\alpha_m : |\Delta c_k^*(\alpha_m) - \Delta c_k^*(\alpha_{m-1})| < 0.05\tau_k, \\ m > \arg\max_m \Delta c_k^*(\alpha_m)\}. \quad (3.1)$$

Figure 2 illustrates this process for determining the optimal threshold for segmenting cell objects from the background. While the first difference $\Delta c_k(\alpha_m)$ measures the immediate change in the number of pixels above each threshold, the second difference more precisely captures the stabilization process of these changes. Initially, at extremely small thresholds, almost all pixels are classified as foreground or cell objects, leading to minimal changes in pixel counts between threshold values, shown as A in Figure 2. However, once the threshold starts to intersect the main background distribution, even small increments can trigger substantial variations, resulting in high second differences. As the threshold increases, the second difference reaches its maximum value at B in Figure 2 and then becomes erratic between B and C in Figure 2. This erratic behavior occurs because, in this transitional range, as the predictive mean values near the decision boundary are very similar, even minor changes in the threshold result in disproportionately large shifts in the number of pixels classified as foreground or cell objects. Beyond this range, the second difference begins to stabilize and falls below a predetermined tolerance level show as E in Figure 2, indicating that further increases yield only negligible changes in segmentation. Any further increase in the threshold results in over-segmentation.

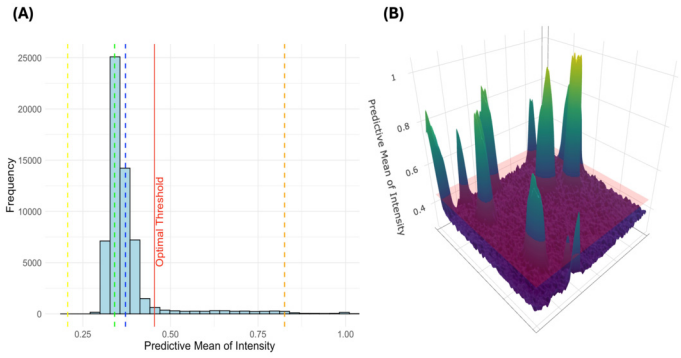


Figure 3: (A) Frequency of the predictive mean of the intensity values over all pixels from the same microscopy image shown in Figure 2 F. The optimal threshold is annotated and lies right after the bulk of the background pixel values. The thresholds that generate images A, B, C, D, and E from Figure 2 are represented as vertical lines with the same color. All pixel intensity values less than the optimal threshold are background pixels and have a symmetric distribution. (B) The predictive mean is shown for each pixel with the optimal threshold plotted as the horizontal plane.

Panel (A) in Figure 3 plots the location of different threshold values in the distribution of pixel intensity values. The optimal threshold selected by the algorithm lies on the right side of the mode. Panel (B) of Figure 3 plots a horizontal plane of the optimal threshold value. A threshold value much smaller than this value misclassifies the background as cell objects, whereas a threshold value much larger than the threshold value misses some small cell objects with low intensity peaks.

After estimating the threshold value α_k^*, we construct a binary image $\mathbf{B}_k^*$ for each cropped image at each pixel:

$$B_k^*(\mathbf{x}_{i,j}) = \begin{cases} 1 & \text{if } F_k^*(\mathbf{x}_{i,j}) > \alpha_k^* \max(\mathbf{F}_k^*) \\ 0 & \text{otherwise} \end{cases}$$

for $k = 1, \ldots, \tilde{K}$. We complete panel (C) in Figure 1 by generating binary cell masks for all sub-images. To address a rare scenario with a higher-than-expected object count, which only happened once for all sub-images, we implemented a re-thresholding step discussed in Section S3 in the Supplementary Material. Finally, each object group is separated and detected via flood fill [4], as described in Section S4 in the Supplementary Material.

3.3 Labeling Cell Masks by the Watershed Algorithm

After obtaining the binary image, we apply the `watershed` algorithm to separate cell objects from EBImage package in R [35], which is particularly useful for segmenting objects connecting to each other. The watershed algorithm creates a topological representation of an image based on a distance map, which is derived from the binary image representing foreground objects (cells). The distance map assigns a value to each foreground pixel based on its Euclidean distance to the nearest background pixel; higher values represent larger distances from the background. These distance values act as heights in a topological landscape, providing a basis for the watershed function to simulate water through. The negative distance map is then calculated by taking the negative of the distance map. Examples of the binary image and negative distance map are provided in Section S5 in the Supplementary Material.

The local minima of the negative distance map generally correspond to the centers of cell objects and are used as the starting point for the "flooding" process in the watershed algorithm. The water fills the cell object from these local minima as the negative distance map is dunked into water, so the cell object acts as a catch basin as the map floods. As water from different starting points meets, watershed lines are formed, which delineate boundaries for distinct binary cell objects. Additionally, whenever the water from a catch basin meets a background pixel, a watershed line is formed [55]. These watershed lines separate and create distinct cell objects, and these cell objects can be assigned unique labels.

An example of the watershed algorithm at distinct water levels is provided in Figure 4 with two cells denoted by d_1 and d_2. The top row of Figure 4 plots the cell images where the negative heights of a slice over the red line are plotted in the bottom row. Figure 4 (A) shows depths for the two local minima for the 1D example are $D(d_1) = -14.14$ and $D(d_2) = -14.21$. Each height acts as a water level, and the local minima act as seed points for water to flow through into each valley or catch basin. When the water level rises to touch a local minimum, the water is given the same label as the local minimum. As the water level rises, all unidentified pixels at or below the water level with neighboring labeled pixels are given the same label.

Figure 4 (B) shows the water levels at -13. Each catch basin is partially filled, as the water level is higher than the two local minima. As the water in each valley is still well separated, all emerged pixels in each valley are either labeled corresponding with d_1 or d_2, with no neighboring pixels containing a different label.

At water level around -10.2, shown in Figure 4 (C), the water from the two different basins begins to touch. The water from the two catch basins corresponding to d_1 and d_2 meet at the unlabeled pixel p_0 with a position close to 14, plotted in the x-coordinate. The Euclidean distances between p_0 and the local minima are utilized to determine its assignment, and a watershed line is formed. The Euclidean distance between p_0 and d_2 (≈ 5.01 units) is smaller than the distance between p_0 and d_1 (≈ 5.62 units), so p_0 is grouped with cell 2. The watershed line will serve as the boundary between the two cells.

At the water level 0, the objects are completely segmented, shown in Figure 4 (D). The watershed operation

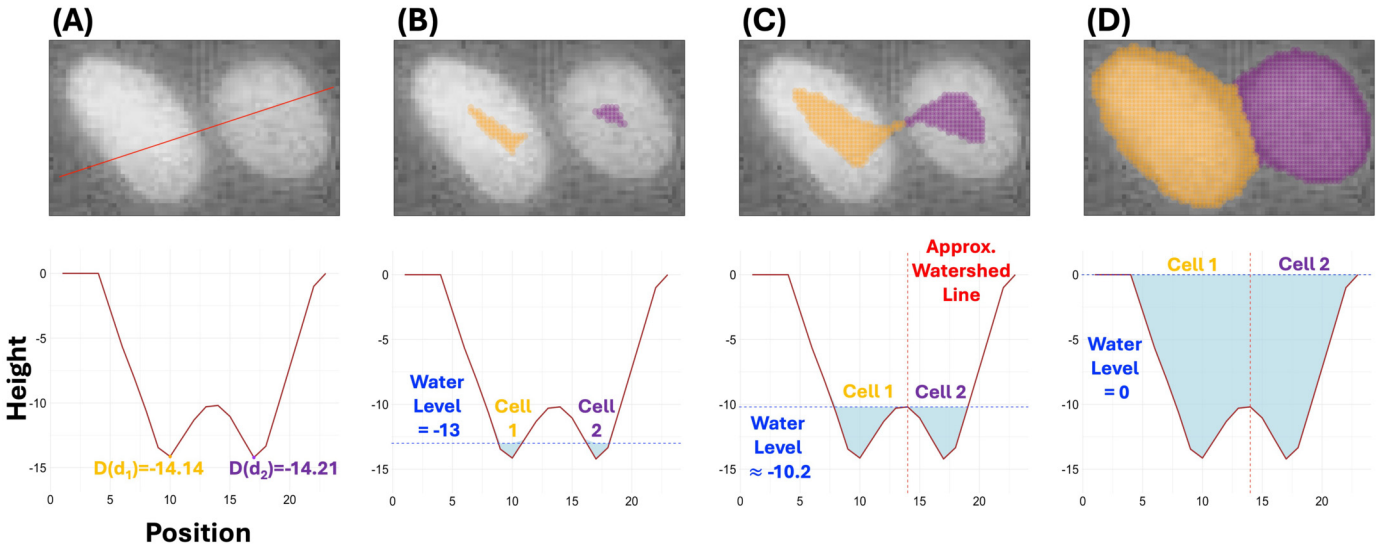


Figure 4: An image of two cell nuclei (upper row) and heights of negative distances to the nearest background pixels (lower row). (A) Heights of negative distances along the red straight line in the upper panel are plotted in the lower panel before the watershed algorithm starts. (B) At water level $= -13$, both catch basins are partially filled, as both local minima are less than the current water level. The two separate water sources have unique labels and are not yet touching. (C) At water level at around -10.2, the water sources from the two catch basins flow into each other and the watershed line is formed at water level at around -10.2. (D) At water level $= 0$, the cell objects are filled with water, and the watershed operation is complete. Each cell is labeled and separated.

then halts, as this height is used as a stopping point since all cell objects have negative heights in the negative distance map. Additional details of the watershed algorithm, including definitions and edge cases, can be found in [55].

Once all objects are segmented by watershed, the average foreground object pixel count is calculated. To filter out any noise that may have been included as a foreground object, all objects with pixel counts less than or equal to 15% of the average are removed. If an object is touching the image boundary, the threshold count for removal is reduced to 5% of the average, as some objects could be partially cut off during imaging.

4. NUMERICAL STUDIES OF IMAGE DENOISING

In this section, we numerically compare our fast GPs with several alternative methods by simulated experiments. The comparison includes the fast GP of a Matérn kernel with roughness parameter being 2.5 in Equation (2.3) and the exponential kernels $K_l(d_l) = \exp(-d_l/\gamma_l)$ where d_l is the distance of the lth coordinate of the input and γ_l is a range parameter, for $l = 1, 2$. We also include three other alternative methods, the principal component analysis (PCA) [3], fast algorithm of multivariate Ornstein-Uhlenbeck processes (FMOU) [27], and dynamic mode decomposition (DMD) [45]. PCA is commonly used in image denoising by projecting image data onto low-dimensional spaces spanned by the leading eigenvectors of the data covariance matrix. FMOU provides a fast expectation-maximization (E-M) algorithm for parameter estimation in a latent factor model $\mathbf{Y} = \mathbf{U}_0\mathbf{Z} + \boldsymbol{\epsilon}$, where $\mathbf{U}_0$ is an orthogonal factor loading matrix and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_d]^T$ consists of d independent latent processes, each modeled as an Ornstein-Uhlenbeck process with distinct correlation and variance parameters, and $\boldsymbol{\epsilon}$ are independent Gaussian noises. In each E-M iteration, parameters are updated using closed-form expressions. DMD is popular in video processing to extract the spatiotemporal structure in nonlinear dynamical systems of high-dimensions. It assumes a linear transition matrix $\mathbf{A}$, estimated as $\hat{\mathbf{A}} = \text{argmin}_{\mathbf{A}} ||\mathbf{Y}_{2:n_2} - \mathbf{A}\mathbf{Y}_{1:n_2-1}||$, where $\mathbf{Y}_{2:n_2}$ and $\mathbf{Y}_{1:n_2-1}$ represent the last and first n_2-1 columns of $\mathbf{Y}$, respectively. An efficient algorithm, known as exact DMD [54], is proposed to identify the leading eigenpairs of $\mathbf{A}$ with computational complexity of $\min(\mathcal{O}(n_1 n_2^2), \mathcal{O}(n_1^2 n_2))$.

We study two examples. In the first example, we studied two scenarios, where the means of the observations follow the Branin function [36] and the linear diffusion equation [31], respectively. In the second example, the means of the observations are images of cell nuclei and whole cells.

Example 1. *In this example, we use the Branin function and linear diffusion equation to generate the mean of the images.*

1. *(Branin function) The mean of the observation is defined by the Branin function*

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 +$$

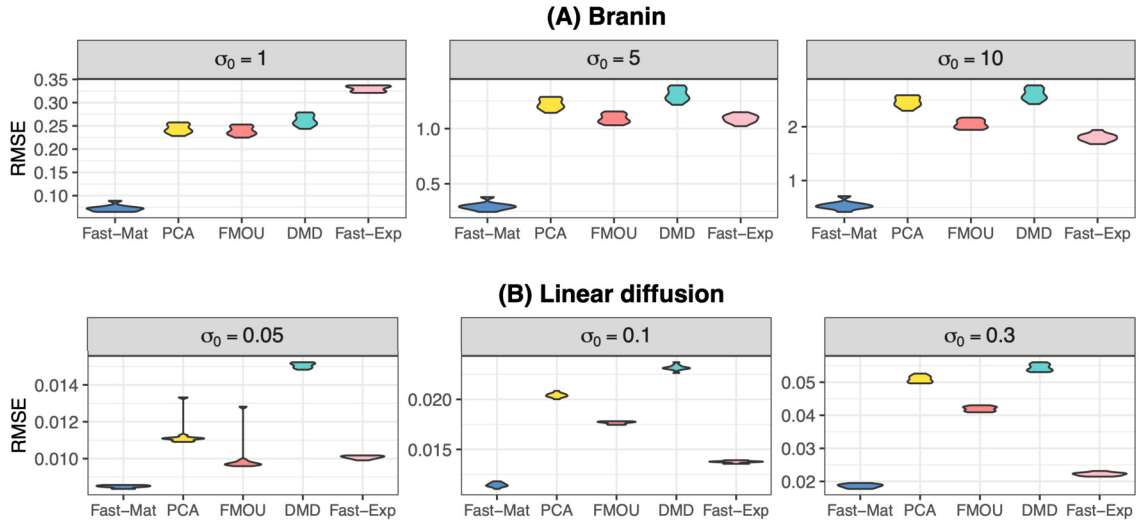


Figure 5: Violin plots of RMSE for five methods applied to the Branin function and the linear diffusion equation across various noise levels. Each experiment is repeated 10 times. The Fast-Mat and Fast-Exp represent the fast GPs with Matérn kernels in Equation (2.3) and exponential kernels, respectively.

$$s(1-t)\cos(x_1)+s,$$

where $x_1 \in [-5, 10]$ *and* $x_2 \in [0, 15]$. *The default parameter values are:* $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$ *and* $t = \frac{1}{8\pi}$. *The input domain is discretized into a uniform grid with* $n_1 = 100$ *points along the* x_1*-axis and* $n_2 = 100$ *points along the* x_2*-axis. The observations are generated by* $y(x_1, x_2) = f(x_1, x_2) + \epsilon$, *where* ϵ *is an independent Gaussian noise with the noise variance* $\sigma_0^2 \in \{1^2, 5^2, 10^2\}$.

2. *(Linear diffusion) The mean of the observation is governed by the partial differential equation*

$$\frac{\partial f(x,t)}{\partial t} = D\frac{\partial^2 f(x,t)}{\partial x^2},$$

where $f(x, t)$ *represents the concentration of the diffusing material at location* x *and time* t, *and* D *is the diffusion coefficient. We set* $D = 1$ *and discretize the spatial domain* $[0, 1]$ *into* 200 *equally spaced grid points. The initial condition is set as* $f(x, 0) = 0$, *with a boundary condition at one end, maintaining a constant external concentration of 1. The signal is simulated over the time interval* $t \in [0, 0.2]$ *using* 200 *time steps, computed with a numerical solver [49]. The observations are generated by* $y(x, t) = f(x, t) + \epsilon$, *where* ϵ *is an independent Gaussian noise. We evaluate the model performance under three configurations with noise variances* $\sigma_0^2 \in \{0.05^2, 0.1^2, 0.3^2\}$.

Example 2 (Cell images)**.** *We consider two noisy cell microscopy images [32]: one for cell nuclei and the other for the whole cell. Independent Gaussian noise with* $\sigma_0^2 \in \{0.1^2, 0.3^2, 0.5^2\}$ *is added to each image to generate the observations.*

To compare the model performance in noise filtering, we consider the error of the predictive mean of the noisy observations, quantified by the root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}(\hat{y}(\mathbf{x}_{i,j}) - \mathbb{E}[y(\mathbf{x}_{i,j})])^2}{n_1 n_2}},$$

where $\hat{y}(\mathbf{x}_{i,j})$ is the estimated mean of $y(\mathbf{x}_{i,j})$.

Figure 5 presents the RMSE in estimating the mean, based on the data-generating processes described in Example 1. Our fast GPs with the Matérn kernel in Equation (2.3) consistently achieve the highest accuracy across varying noise levels in both scenarios. This is because GPs with separable kernels capture spatial dependencies in both input directions, which are crucial for modeling images. In contrast, PCA implicitly assumes that the prior in one direction is independent, as the corresponding probabilistic model assumes that the latent factors are distributed as a standard Gaussian distribution [52]. Though DMD and FMOU assume the latent factors are from Gaussian processes, which capture output correlation over one input dimension, the latent factor loading matrix is estimated without modeling the correlation over the other input by a kernel function. In comparison, Fast GPs with Matérn kernel directly model the correlation over two inputs through a product kernel, which induces better predictions of the signals. Additionally, since DMD implicitly assumes a noise-free model [18], leading to degraded performance as noise variance increases. Figure S6 in the Supplementary Material shows the predictive signal by Fast GPs with Matérn kernel from noisy observations generated by the Branin function and the linear diffusion equation, which demonstrates close alignment with the ground truth.

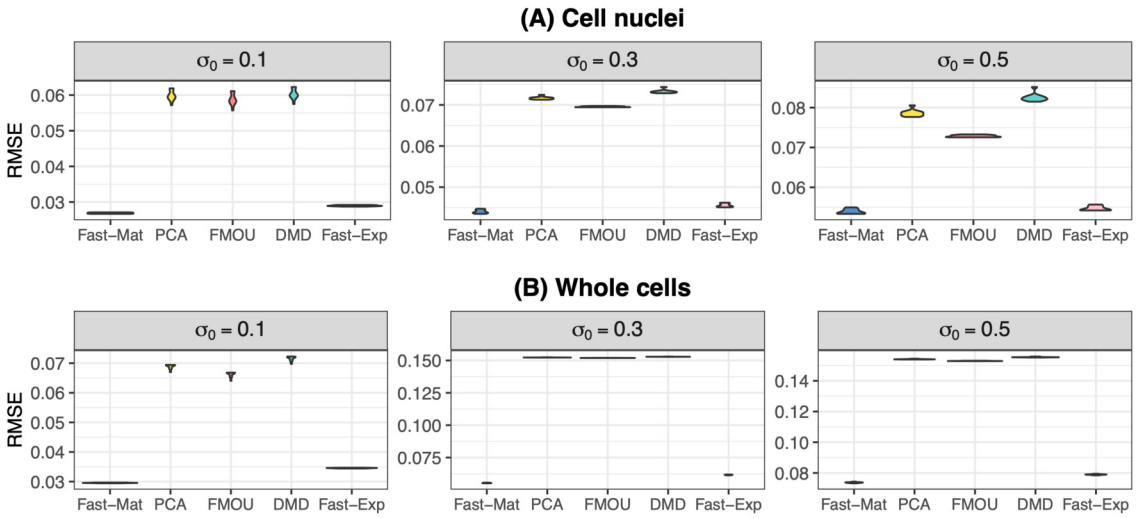


Figure 6: Violin plots of RMSE for five methods applied to images of noisy cell nuclei and whole cells across various noise levels. Each experiment is repeated 10 times. The Fast-Mat and Fast-Exp represent the fast GPs with Matérn kernels in Equation (2.3) and exponential kernels, respectively.

Figure 6 compares the accuracy of five approaches in denoising cell nuclei and whole-cell images in Example 2. The fast GPs with Matérn and exponential kernels significantly outperform PCA, FMOU and DMD across all settings, owing to their ability to estimate correlation parameters in both directions. Figure 7 plots the signal, noisy observations, and predictive mean from the fast GPs with a Matérn kernel in Equation (2.3) for denoising the noisy cell images. The predictive mean is close to the latent mean of the observations.

Additionally, we compare the computational cost of evaluating the profile likelihood by direct computation and fast computation. Simulations are conducted across varying sample sizes, ranging from 10^2 to 80^2. As shown in Figure 8, the fast algorithm in Equation (2.9) significantly reduces computational time compared to the direct approach, particularly for large sample sizes.

5. NUMERICAL STUDIES OF CELL SEGMENTATION

5.1 Criteria and Methods

We use microscopy images of human dermal fibroblasts (hDFs) in [32] for testing different approaches. More details on the experimental conditions and techniques can be found in Section S7 in the Supplementary Material. These images contain both nuclei and cytoplasm channels, and evaluation of the proposed GP-based segmentation method was performed on both channels.

To evaluate the accuracy of the truth and segmentation results, we compare the area of overlap between the detected object and the true object relative to the overall union area of the two objects [39]. Let g represent a ground truth mask and p a predicted mask. The Intersection over Union (IoU) metric between g and p is defined as:

$$\mathrm{IoU}(g,p) = \frac{|g \cap p|}{|g \cup p|}. \tag{5.1}$$

A higher IoU indicates a better match between the predictive mask and true mask of an object, with the perfect match occurring at an IoU of 1. We test several levels of IoU for each method: $\alpha = [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8]$. When $\alpha = 0.6$, for instance, it means that whenever the IoU defined in Equation (5.1) of a cell is larger than 0.6 by a certain method, this object will be classified as a true positive for this method. The range of thresholds allows us to assess image segmentation performance under distinct precision levels for defining true positives.

The average precision metric (AP(α)) at each IoU threshold $\alpha \in [0,1]$ is often used for evaluating the correction of image segmentation [50]. For $\alpha \in [0,1]$, a True Positive (TP) means that the ground truth mask g is matched to a predicted mask p with $\mathrm{IoU}(g,p) \geq \alpha$. Otherwise, unmatched predicted masks are counted as False Positives (FP), and unmatched ground truth masks are counted as False Negatives (FN). The AP(α) is calculated as follows:

$$\mathrm{AP}(\alpha) = \frac{\mathrm{TP}(\alpha)}{\mathrm{TP}(\alpha) + \mathrm{FP}(\alpha) + \mathrm{FN}(\alpha)}. \tag{5.2}$$

We compare ImageJ segmentation and the GP-based unsupervised detection algorithm developed in this work. We followed the conventional processing steps in ImageJ to generate the cell masks [46]. All input images are converted to grayscale. To detect the foreground and background pixels, a threshold was set for each image using the default methods.

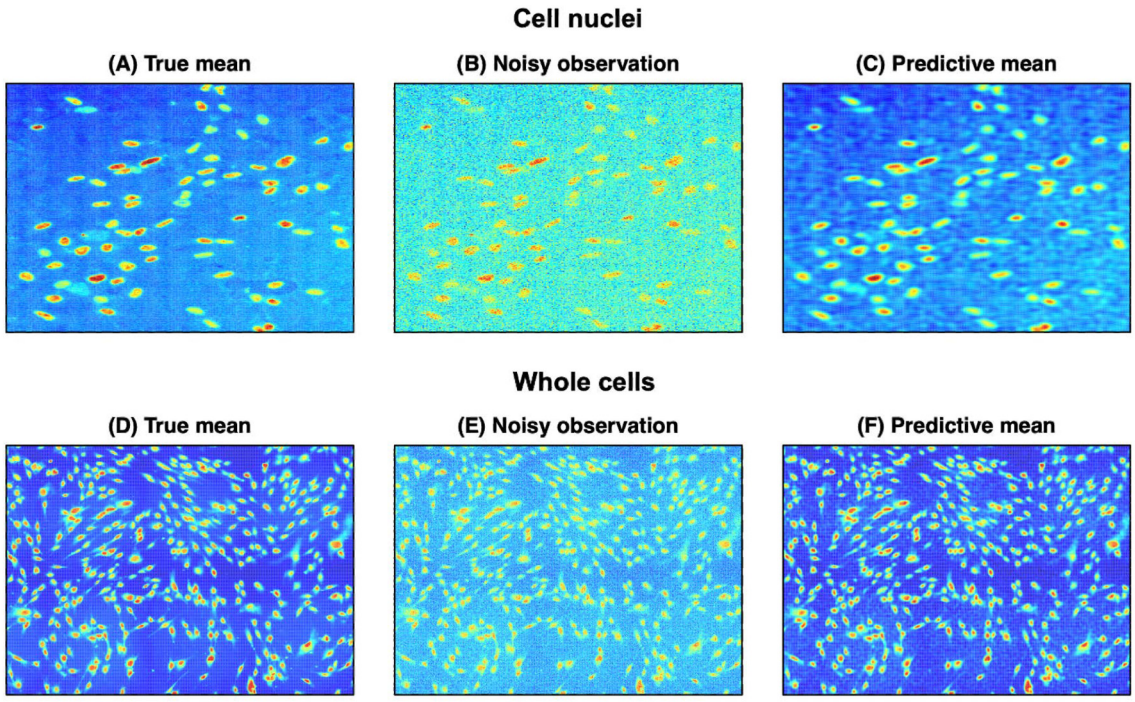


Figure 7: (A) True mean of cell nuclei. (B) Noisy observation of cell nuclei ($\sigma_0 = 0.1$). (C) Predictive mean of cell nuclei by fast GPs on lattice data with Matérn kernels in Equation (2.3). (D) True mean of the whole cell. (E) Noisy observation of the whole cell ($\sigma_0 = 0.1$). (F) Predictive mean of the whole cell by Fast GPs on lattice data with Matérn kernels.

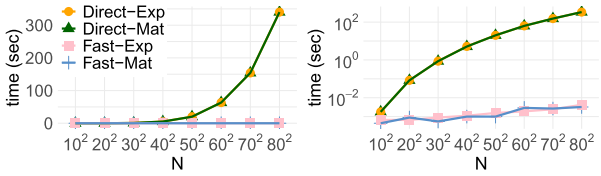


Figure 8: Comparison of the computational time in seconds for evaluating the profile likelihood of $\mathbf{Y}_v$ using direct and fast computation under varying sample sizes N. The direct computation with the exponential kernel (Direct-Exp) is plotted as orange dots; the direct computation with the Matérn kernels in Equation (2.3) (Direct-Mat) is plotted as green triangles; 3) the fast computation with the exponential kernel (Fast-Exp) is plotted as pink squares; The fast computation with the Matérn kernels in Equation (2.3) (Fast-Mat) is plotted as blue crosses. The left panel shows the computational time in seconds (original scale), while the right panel displays the logarithmic scale.

To reduce noises in ImageJ, a despeckle operation, which applies a 3×3 median filter, was then applied to each image. A watershed algorithm was used to ensure better separation of connecting cells. Finally, the cell masks were generated. The cell size is selected to be larger than 30 pixels. Then, the binary image and the labeled mask can be generated in ImageJ. A detailed description of each step in this process for ImageJ can be found in Section S8 in the Supplementary Material.

5.2 Image Segmentation Results for Cell Nuclei

We first use five images of cell nuclei to test different methods. We choose a wide range of image sizes to test methods for images with different sizes. The smallest dimension of these images is 374×250, and the largest image is 962×1128. The experimental details of these images are shown in Table S1 in the Supplementary Material.

In panel (A) of Figure 9, we plot the AP(α) between the GP-based segmentation and ImageJ segmentation methods by solid lines and dashed lines, respectively, for 5 test images of nuclei at distinct threshold α. For any image and threshold level, the solid line is almost always above the dashed line with the same color, as the GP-based method consistently achieves higher AP than ImageJ at any threshold level α. The performance gap becomes more obvious at intermediate IoU thresholds between 0.6 to 0.7, which means that the GP-based method performs better at recovering the overall structure of the object than ImageJ. The AP score of both methods drops at a high IoU threshold, as the annotated object may not be fully accurate, which can affect defining the truth. Panel (B) in Figure 9 shows the distribution of these AP scores from GPs and ImageJ segmentation methods averaged at each threshold. Similar to the results

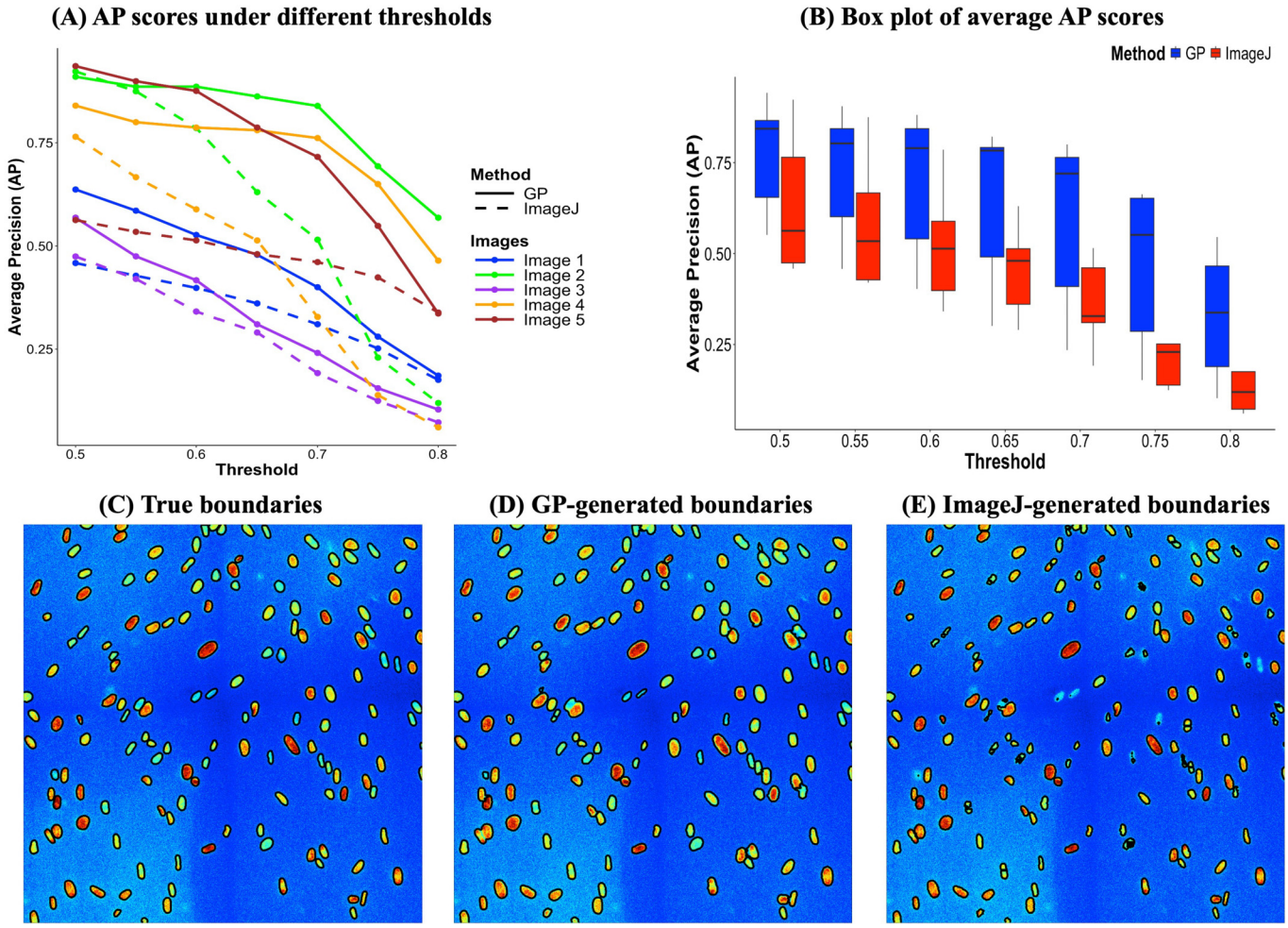


Figure 9: Results for segmenting cell nuclei. (A) AP scores for GP-based segmentation and ImageJ across different thresholds. (B) Boxplots of AP scores for each method. These plots are created using the APs for each of the 5 images using each method across different thresholds. (C) True boundaries of the fifth test image with 1024×1024 pixels. (D) Boundaries generated by the GP-based method for the fifth test image. (E) Boundaries generated using ImageJ for the fifth test image.

in panel (A), the GP-based method achieves higher median AP scores across all thresholds. The difference between the GP-based method and ImageJ is more pronounced at a high threshold level, such as $\alpha = 0.75$ or $\alpha = 0.8$, where the boxes of two colors nearly have no overlap, indicating that GP-based method better capture the fine-scale boundary details than ImageJ.

The true annotated boundaries, those generated by the GP-based method, and those generated by ImageJ of the fifth test image are plotted in Figure 9 (C)-(E). The GP-based image segmentation method generally produces smoother, more continuous boundaries capturing the actual cell nuclei, even in cases where cells were closely clustered. In contrast, the ImageJ segmentation method sometimes produces fragmented boundaries and erroneously segmented individual nuclei into multiple smaller regions, especially in regions where the variations of pixel intensity are large. Additionally, ImageJ fails to detect multiple cells entirely. Further image refinement may be achieved using additional plugins or filters from ImageJ. However, this process would require extensive human intervention for trial and error, which would substantially increase the image processing time.

5.3 Image Segmentation Results for Whole Cells

We compare the performance of the GP-based segmentation method and the ImageJ method for five whole-cell images. The smallest image is 602×600, and the largest is 1000×1000.

Panel (A) of Figure 10 shows the AP with different thresholds by the two methods for five images of whole cells. The variation of AP by both methods for the five images of whole cells is larger than the five images of the nuclei images. This is expected as segmenting whole cells is much more challenging due to the distinct shapes of the whole-cell

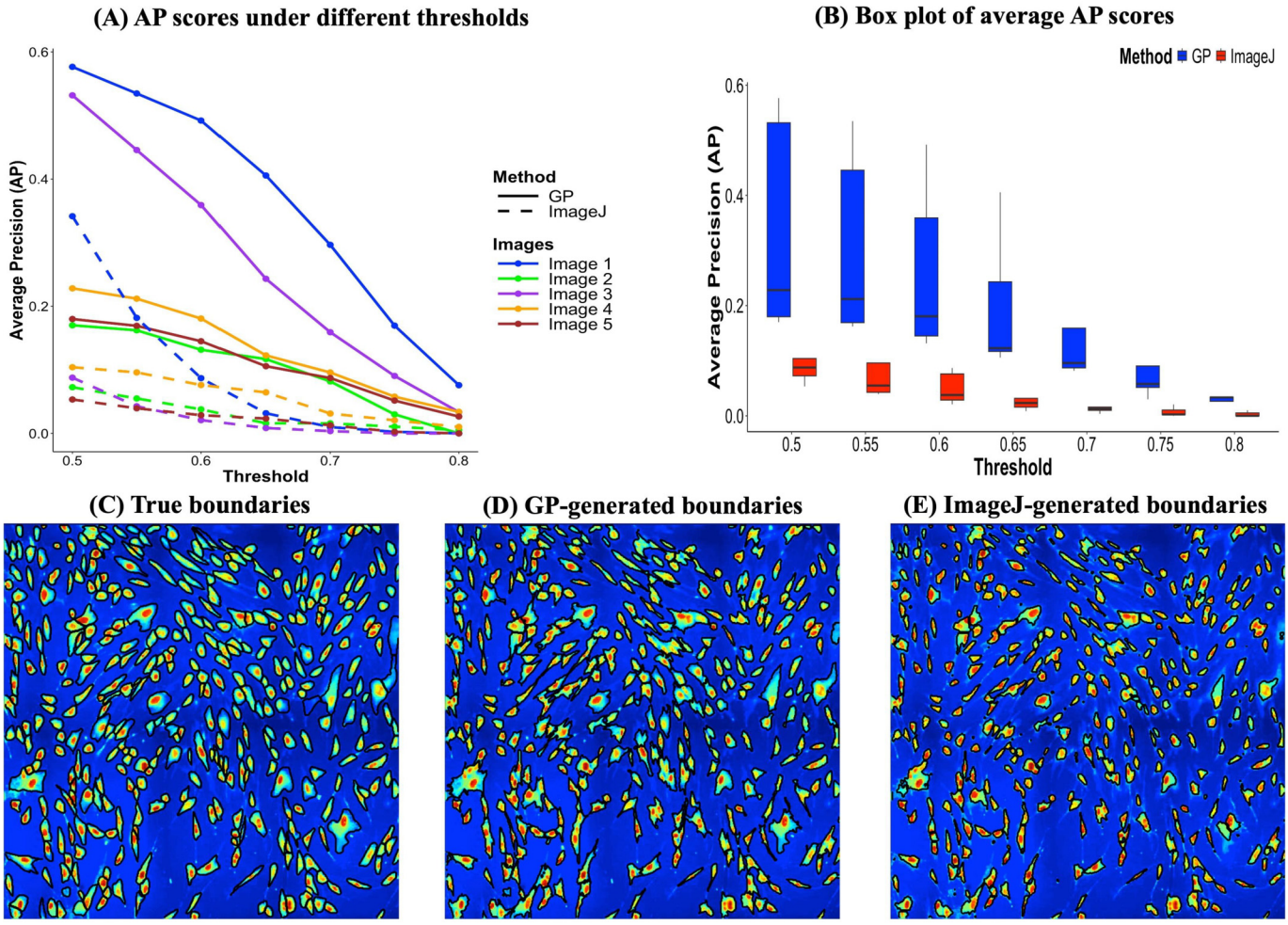


Figure 10: Results for segmenting whole cells. (A) AP scores for GP-based segmentation and ImageJ across different thresholds. (B) Boxplots of AP scores for each method. (C) True boundaries of the first test image with 602×600 pixels. (D) Boundaries generated by the GP-based method of the first test image. (E) Same for panel (D) but generated using ImageJ.

images and more objects connecting to each other. Nonetheless, the solid line representing the AP for the GP-based method is again almost always above the dashed line representing the AP of the ImageJ segmentation results with the same color for all thresholds, meaning that the GP method consistently achieves higher AP scores than ImageJ. Both methods show much lower precision at a smaller IoU threshold, as correctly identifying irregular boundary shapes in connecting whole cells is extremely hard to capture. Such boundaries also pose challenges for humans to annotate, and for supervised methods to segment, even when annotated data are available.

Panel (B) of Figure 10 shows the distribution of the AP scores between the two methods. Compared to the segmentation results of the cell nuclei, the improvement by the GP-based segmentation methods is more pronounced for images of whole cells. The variation of AP by the GP-based segmentation is larger for whole-cell images than the ones for the images of nuclei, due to the difficulty in identifying the whole-cell shape either by statistical learning methods or by humans.

The ground truth boundaries, the GP-based method generated boundaries, and the ImageJ generated boundaries of the first text image are given in panels (C)-(E) in Figure 10. The GP-based method generally produces boundaries that better fit the actual cell shapes, preserving the irregular shapes of the cell cytoplasm. Although the GP-based method generally better preserves the true cell shapes, cell objects were also often grouped together. In contrast, ImageJ segmentation often produces fragmented boundaries and splits individual cell shapes into multiple smaller regions.

6. CONCLUDING REMARKS

In this study, we have presented a scalable cell image segmentation method for both the cell nucleus and cytoplasm. By leveraging fast Gaussian processes, automated thresholding and watershed operations, we have shown that the process is not only more automated, but the results are more precise than other unsupervised segmentation methods, such as ImageJ. A key strength of the proposed segmentation algorithm is that it does not rely on parameter tuning or labeled training datasets. The results of our unsupervised method demonstrate versatility across different cell channels, making it an appealing option for more general segmentation tasks beyond cell segmentation.

Several future directions are worth exploring. First, modern microscopes provide time-lapse videos for tracking the changes in cellular behavior over time. A future goal is to integrate our segmentation methods with particle tracking and linking algorithms that solve a linear assignment problem [22]. As our approach does not restrict the shape of the object, it can be extended for image segmentation with objects undergoing dynamical changes, including microscopy images for the nucleation and growth process in crystallization [56] and protein dynamics [26]. Second, when labeled image data are available, it is worthwhile to explore how these data can be used for constraining the shapes of the objects to improve estimation and simultaneously enable segmenting new objects not in the database. Third, the majority of cell segmentation methods are developed for 2D microscopy images. It is of interest to develop cell segmentation methods for microscopy that capture cell behaviors in a 3D environment, such as the orientational order and alignment [21]. Similar to some popular cell segmentation tools, such as Cellpose [50], our approach aims to segment images that contain the same type of cell. A future direction is to leverage techniques in tools such as CellProfiler [5] to segment images that contain multiple types of cells. Furthermore, the uncertainty is often not quantified by the image segmentation methods, as the segmentation process often contains multiple steps, and the uncertainty from the convolutional neural network is an open question. As we model the image data with a probabilistic model, a future goal is to propagate the uncertainty throughout the analysis for uncertainty quantification.

APPENDIX A. PROOF OF LEMMA 1

Proof. We first prove part 1 of the Lemma 1. Given the eigendecompositions $\mathbf{R}_l = \mathbf{U}_l\mathbf{\Lambda}_l\mathbf{U}_l^T$, where $\mathbf{U}_l \in \mathbb{R}^{n_l\times n_l}$ for $l = 1, 2$. Then the Kronecker product $\tilde{\mathbf{R}} = \mathbf{R}_2 \otimes \mathbf{R}_1 + \eta\mathbf{I}_N$ can be written as,

$$\begin{aligned}\tilde{\mathbf{R}} =&\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^T \otimes \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^T + \eta\mathbf{I}_N\\ =&[(\mathbf{U}_2\mathbf{\Lambda}_2) \otimes (\mathbf{U}_1\mathbf{\Lambda}_1)](\mathbf{U}_2^T \otimes \mathbf{U}_1^T) + \eta\mathbf{I}_N\\ =&(\mathbf{U}_2 \otimes \mathbf{U}_1)(\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1)(\mathbf{U}_2^T \otimes \mathbf{U}_1^T) + \eta\mathbf{I}_N\\ =&\mathbf{Q}\tilde{\mathbf{\Lambda}}\mathbf{Q}^T,\end{aligned}$$

where $\mathbf{Q} = \mathbf{U}_2 \otimes \mathbf{U}_1$ is an orthogonal matrix, and $\tilde{\mathbf{\Lambda}} = (\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1 + \eta\mathbf{I}_N)$ is a diagonal matrix. Thus,

$$|\tilde{\mathbf{R}}| = \prod_{i=1}^{n_1}\prod_{j=1}^{n_2}(\lambda_{i,1}\lambda_{j,2} + \eta). \tag{A.1}$$

Denote $\mathbf{1}_N = \mathbf{1}_{n_2} \otimes \mathbf{1}_{n_1}$ and we have

$$\hat{\mu} = \frac{\mathbf{1}_N^T\tilde{\mathbf{R}}^{-1}\mathbf{y}_v}{\mathbf{1}_N^T\tilde{\mathbf{R}}^{-1}\mathbf{1}_N}. \tag{A.2}$$

The numerator of Equation (A.2) can be written as

$$\begin{aligned}&\mathbf{1}_N^T\tilde{\mathbf{R}}^{-1}\mathbf{y}_v\\ =&\mathbf{1}_N^T(\mathbf{Q}\tilde{\mathbf{\Lambda}}\mathbf{Q}^T)^{-1}\mathbf{y}_v\\ =&\mathbf{1}_N^T(\mathbf{U}_2 \otimes \mathbf{U}_1)\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2 \otimes \mathbf{U}_1)^T\mathrm{Vec}(\mathbf{Y})\\ =&[(\mathbf{1}_{n_2}^T\mathbf{U}_2) \otimes (\mathbf{1}_{n_1}^T\mathbf{U}_1)]\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\mathrm{Vec}(\mathbf{Y})\\ =&(\tilde{\mathbf{u}}_2^T \otimes \tilde{\mathbf{u}}_1^T)\tilde{\mathbf{\Lambda}}^{-1}\mathrm{Vec}(\mathbf{U}_1^T\mathbf{Y}\mathbf{U}_2)\\ =&\sum_{j=1}^{n_2}\sum_{i=1}^{n_1}\frac{\tilde{u}_{i,1}\tilde{Y}_{i,j,0}\tilde{u}_{j,2}}{\lambda_{i,1}\lambda_{j,2}+\eta}.\end{aligned}$$

where $\tilde{Y}_{i,j,0}$ is the (i,j)th entry of the $n_1 \times n_2$ matrix $\mathbf{U}_1^T\mathbf{Y}\mathbf{U}_2$, $\tilde{u}_{i,1}$ is the ith term of $\tilde{\mathbf{u}}_1 = \mathbf{U}_1^T\mathbf{1}_{n_1}$, and $\tilde{u}_{j,2}$ is the jth term of $\tilde{\mathbf{u}}_2 = \mathbf{U}_2^T\mathbf{1}_{n_2}$, for $l = 1, 2$, $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$.

Similarly, the denominator of Equation (A.2) can be written as

$$\begin{aligned}&\mathbf{1}_N^T\tilde{\mathbf{R}}^{-1}\mathbf{1}_N\\ =&\mathbf{1}_N^T(\mathbf{U}_2 \otimes \mathbf{U}_1)\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\mathbf{1}_N\\ =&[(\mathbf{1}_{n_2}^T\mathbf{U}_2) \otimes (\mathbf{1}_{n_1}^T\mathbf{U}_1)]\tilde{\mathbf{\Lambda}}^{-1}[(\mathbf{U}_2^T\mathbf{1}_{n_2}) \otimes (\mathbf{U}_1^T\mathbf{1}_{n_1})]\\ =&(\tilde{\mathbf{u}}_2^T \otimes \tilde{\mathbf{u}}_1^T)\tilde{\mathbf{\Lambda}}^{-1}(\tilde{\mathbf{u}}_2 \otimes \tilde{\mathbf{u}}_1)\\ =&\sum_{j=1}^{n_2}\sum_{i=1}^{n_1}\frac{\tilde{u}_{i,1}^2\tilde{u}_{j,2}^2}{\lambda_{i,1}\lambda_{j,2}+\eta}.\end{aligned}$$

The term S^2 in the log likelihood function in Equation (2.4) follows

$$\begin{aligned}S^2 =&(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)^T\tilde{\mathbf{R}}^{-1}(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)\\ =&(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)^T\mathbf{Q}\tilde{\mathbf{\Lambda}}^{-1}\mathbf{Q}^T(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)\\ =&[\mathbf{Q}^T(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)]^T\tilde{\mathbf{\Lambda}}^{-1}\mathbf{Q}^T(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N)\\ =&[(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\mathrm{Vec}(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1\times n_2})]^T\\ &\tilde{\mathbf{\Lambda}}^{-1} \times (\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\mathrm{Vec}(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1\times n_2})\\ =&[\mathrm{Vec}(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1\times n_2})\mathbf{U}_2)]^T\\ &\tilde{\mathbf{\Lambda}}^{-1}\mathrm{Vec}(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1\times n_2})\mathbf{U}_2)\\ =&\tilde{\mathbf{y}}_v^T\tilde{\mathbf{\Lambda}}^{-1}\tilde{\mathbf{y}}_v\end{aligned} \tag{A.3}$$

$$= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \frac{\tilde{Y}_{i,j}^2}{\lambda_{i,1}\lambda_{j,2} + \eta}, \tag{A.4}$$

where $\tilde{\mathbf{y}}_v = \text{Vec}(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1 \times n_2})\mathbf{U}_2)$. By Equations (2.4), (A.1) and (A.4), we have Equation (2.9).

For part 2 of the lemma, we first calculate the predictive mean vector below

$$\begin{aligned}
\mathbf{f}_v^* =& \hat{\mu}\mathbf{1}_N + \mathbf{R}\tilde{\mathbf{R}}^{-1}(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N) \\
=& \hat{\mu}\mathbf{1}_N + (\mathbf{U}_2 \otimes \mathbf{U}_1)(\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1)(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\times \\
& (\mathbf{U}_2 \otimes \mathbf{U}_1)\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)(\mathbf{y}_v - \hat{\mu}\mathbf{1}_N) \\
=& \hat{\mu}\mathbf{1}_N + (\mathbf{U}_2 \otimes \mathbf{U}_1)[(\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1)\tilde{\mathbf{\Lambda}}^{-1}]\times \\
& \text{Vec}(\mathbf{U}_1^T(\mathbf{Y} - \hat{\mu}\mathbf{1}_{n_1 \times n_2})\mathbf{U}_2) \\
=& \hat{\mu}\mathbf{1}_N + (\mathbf{U}_2 \otimes \mathbf{U}_1)[(\mathbf{\Lambda}_2 \otimes \mathbf{\Lambda}_1)\tilde{\mathbf{\Lambda}}^{-1}]\tilde{\mathbf{y}}_v \\
=& \hat{\mu}\mathbf{1}_N + (\mathbf{U}_2 \otimes \mathbf{U}_1)\text{Vec}\left(\mathbf{\Lambda}_1\tilde{\mathbf{Y}}_0\mathbf{\Lambda}_2\right) \\
=& \hat{\mu}\mathbf{1}_N + \text{Vec}\left(\mathbf{U}_1\mathbf{\Lambda}_1\tilde{\mathbf{Y}}_0\mathbf{\Lambda}_2\mathbf{U}_2^T\right),
\end{aligned}$$

where $\text{Vec}(\tilde{\mathbf{Y}}_0) = \tilde{\mathbf{\Lambda}}^{-1}\tilde{\mathbf{y}}_v$.

To compute the predictive variance, the term $\mathbf{R}\tilde{\mathbf{R}}^{-1}\mathbf{R}$ follows

$$\begin{aligned}
\mathbf{R}\tilde{\mathbf{R}}^{-1}\mathbf{R} =& (\mathbf{R}_2 \otimes \mathbf{R}_1)(\mathbf{U}_2 \otimes \mathbf{U}_1)\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\times \\
& (\mathbf{R}_2 \otimes \mathbf{R}_1) \\
=& (\mathbf{R}_2\mathbf{U}_2 \otimes \mathbf{R}_1\mathbf{U}_1)\tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T\mathbf{R}_2 \otimes \mathbf{U}_1^T\mathbf{R}_1)
\end{aligned}$$

The $t = i + (j-1)n_1$ diagonal term of $\hat{\sigma}^2\mathbf{R}^*$ is the predictive variance at pixel location (i, j), which follows

$$\begin{aligned}
\hat{\sigma}^2 c_{i,j}^* =& \hat{\sigma}^2 - \hat{\sigma}^2(\mathbf{r}_{j,2}^T\mathbf{U}_2 \otimes \mathbf{r}_{i,1}^T\mathbf{U}_1)\times \\
& \tilde{\mathbf{\Lambda}}^{-1}(\mathbf{U}_2^T\mathbf{r}_{j,2} \otimes \mathbf{U}_1^T\mathbf{r}_{i,1}) \\
=& \hat{\sigma}^2\left(1 - \sum_{j'=1}^{n_2}\sum_{i'=1}^{n_1} \frac{\tilde{r}_{i,i',1}^2\tilde{r}_{j,j',2}^2}{\lambda_{i,1}\lambda_{j,2} + \eta}\right),
\end{aligned}$$

where $\tilde{r}_{i,i',1}$ is the i'th term of the vector $\mathbf{r}_{i,1}^T\mathbf{U}_1$ and $\tilde{r}_{j,j',2}$ is the j'th term of the vector $\mathbf{r}_{j,2}^T\mathbf{U}_2$, for $i' = 1, \ldots, n_1$ and $j' = 1, \ldots, n_2$, with $\mathbf{r}_{i,1}^T = (K_1(x_{i,1}, x_{1,1}), \ldots, K_1(x_{i,1}, x_{n_1,1}))^T$ and $\mathbf{r}_{j,2}^T = (K_2(x_{j,2}, x_{1,2}), \ldots, K_2(x_{j,2}, x_{n_2,2}))^T$. □

SUPPLEMENTARY MATERIAL

The supplementary material provides additional details for image segmentation, experiments, and generation of ground truth.

ACKNOWLEDGMENTS

We thank three anonymous referees for their comments that substantially improved this article. The manuscript is available in arXiv: https://doi.org/10.48550/arXiv.2505.18902.

FUNDING

This research is partially supported by the Materials Research Science and Engineering Center (MRSEC, Data Expert Group and IRG-2) by the National Science Foundation under Award No. DMR-2308708 and the Cyberinfrastructure for Sustained Scientific Innovation program by the National Science Foundation under Award No. OAC-2411043.

Accepted 5 January 2026

REFERENCES

[1] Abràmoff, M. D., Magalhães, P. J. and Ram, S. J. (2004). Image processing with ImageJ. *Biophotonics International* **11**(7) 36–42.

[2] Barbazan, J., Pérez-González, C., Gómez-González, M., Dedenon, M., Richon, S., Latorre, E., Serra, M., Mariani, P., Descroix, S., Sens, P. et al. (2023). Cancer-associated fibroblasts actively compress cancer cells and modulate mechanotransduction. *Nature Communications* **14**(1) 6966.

[3] Berkooz, G., Holmes, P. and Lumley, J. L. (1993). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of Fluid Mechanics* **25**(1) 539–575. MR1204279

[4] Bhawnesh, K., Tiwari, U., Kumar, S., Tomer, V. and Kalra, J. (2020). Comparison and performance evaluation of boundary fill and flood fill algorithm. *International Journal of Innovative Technology and Exploring Engineering* **8**. https://doi.org/10.35940/ijitee.L1002.10812S319

[5] Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D. A., Chang, J. H., Lindquist, R. A., Moffat, J. et al. (2006). CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology* **7**(10) 1–11.

[6] Chang, W., Haran, M., Applegate, P. and Pollard, D. (2016). Calibrating an ice sheet model using high-dimensional binary spatial data. *Journal of the American Statistical Association* **111**(513) 57–72. https://doi.org/10.1080/01621459.2015.1108199. MR3494638

[7] Datta, A., Banerjee, S., Finley, A. O. and Gelfand, A. E. (2016). Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association* **111**(514) 800–812. https://doi.org/10.1080/01621459.2015.1044091. MR3538706

[8] Eisenbarth, S. (2019). Dendritic cell subsets in T cell programming: location dictates function. *Nature Reviews Immunology* **19**(2) 89–103.

[9] Eisenhoffer, G. T., Loftus, P. D., Yoshigi, M., Otsuna, H., Chien, C. -B., Morcos, P. A. and Rosenblatt, J. (2012). Crowding induces live cell extrusion to maintain homeostatic cell numbers in epithelia. *Nature* **484**(7395) 546–549.

[10] Fang, X. and Gu, M. (2024). The inverse Kalman filter. *arXiv preprint arXiv:2407.10089*. https://doi.org/10.1093/biomet/asaf054. MR4985298

[11] Ferreira, T. and Rasband, W. (2011). ImageJ user guide. *USA: National Institutes of Health*.

[12] Folkman, J. and Moscona, A. (1978). Role of cell shape in growth control. *Nature* **273**(5661) 345–349.

[13] Gramacy, R. B. and Apley, D. W. (2015). Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics* **24**(2) 561–578. https://doi.org/10.1080/10618600.2014.914442. MR3357395

[14] Gu, M. and Li, H. (2022). Gaussian Orthogonal Latent Factor Processes for Large Incomplete Matrices of Correlated Data. *Bayesian Analysis* **17**(4) 1219–1244. https://doi.org/10.1214/21-ba1295. MR4506027

[15] Gu, M., Fang, X. and Luo, Y. (2023). Data-driven model construction for anisotropic dynamics of active matter. *PRX Life* **1**(1) 013009.
[16] Gu, M., Palomo, J. and Berger, J. O. (2019). RobustGaSP: Robust Gaussian Stochastic Process Emulation in R. *The R Journal* **11**(1) 112–136. https://doi.org/10.32614/RJ-2019-011
[17] Gu, M., Wang, X. and Berger, J. O. (2018). Robust Gaussian stochastic process emulation. *Annals of Statistics* **46**(6A) 3038–3066. https://doi.org/10.1214/17-AOS1648. MR3851764
[18] Gu, M., Lin, Y., Lee, V. C. and Qiu, D. Y. (2024). Probabilistic forecast of nonlinear dynamical systems with uncertainty quantification. *Physica D: Nonlinear Phenomena* **457** 133938. https://doi.org/10.1016/j.physd.2023.133938. MR4660232
[19] Guinness, J. and Fuentes, M. (2017). Circulant embedding of approximate covariances for inference from Gaussian data on large lattices. *Journal of computational and Graphical Statistics* **26**(1) 88–97. https://doi.org/10.1080/10618600.2016.1164534. MR3610410
[20] Handcock, M. S. and Stein, M. L. (1993). A Bayesian analysis of kriging. *Technometrics* **35**(4) 403–410.
[21] Huang, J., Chen, J. and Luo, Y. (2025). Cell-Sheet Shape Transformation by Internally-Driven, Oriented Forces. *Advanced Materials* 2416624.
[22] Jaqaman, K., Loerke, D., Mettlen, M., Kuwata, H., Grinstein, S., Schmid, S. L. and Danuser, G. (2008). Robust single-particle tracking in live-cell time-lapse sequences. *Nature methods* **5**(8) 695–702.
[23] Katzfuss, M. (2017). A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association* **112**(517) 201–214. https://doi.org/10.1080/01621459.2015.1123632. MR3646566
[24] Kaufman, C. G., Schervish, M. J. and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association* **103**(484) 1545–1555. https://doi.org/10.1198/016214508000000959. MR2504203
[25] Khang, A., Barmore, A., Tseropoulos, G., Bera, K., Batan, D. and Anseth, K. S. (2025). Automated prediction of fibroblast phenotypes using mathematical descriptors of cellular features. *Nature Communications* **16**(1) 1–17.
[26] Lee, G., Leech, G., Rust, M. J., Das, M., McGorty, R. J., Ross, J. L. and Robertson-Anderson, R. M. (2021). Myosin-driven actin-microtubule networks exhibit self-organized contractile dynamics. *Sci. Adv.* **7**(6) 4334.
[27] Lin, Y., Liu, X., Segall, P. and Gu, M. (2025). Fast data inversion for high-dimensional dynamical systems from noisy measurements. *arXiv preprint arXiv:2501.01324*.
[28] Lindgren, F., Rue, H. and Lindström, J. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**(4) 423–498. https://doi.org/10.1111/j.1467-9868.2011.00777.x. MR2853727
[29] Liu, D. and Yu, J. (2009). Otsu method and K-means. In *2009 Ninth International Conference on Hybrid Intelligent Systems* **1** 344–349. IEEE.
[30] Liu, W. F. and Chen, C. S. (2007). Cellular and multicellular form and function. *Advanced Drug Delivery Reviews* **59**(13) 1319–1328.
[31] Lu, H. and Tartakovsky, D. M. (2020). Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing* **42**(3) 1639–1662. https://doi.org/10.1137/19M1259948. MR4102719
[32] Luo, Y., Gu, M., Park, M., Fang, X., Kwon, Y., Urueña, J. M., Read de Alaniz, J., Helgeson, M. E., Marchetti, C. M. and Valentine, M. T. (2023). Molecular-scale substrate anisotropy, crowding and division drive collective behaviours in cell monolayers. *Journal of the Royal Society Interface* **20**(204) 20230160.
[33] Meijering, E. (2012). Cell segmentation: 50 years down the road. *IEEE Signal Processing Magazine* **29**(5) 140–145.
[34] Nichele, L., Persichetti, V., Lucidi, M. and Cincotti, G. (2020). Quantitative evaluation of ImageJ thresholding algorithms for microbial cell counting. *OSA Continuum* **3**(6) 1417–1427.
[35] Pau, G., Fuchs, F., Sklyar, O., Boutros, M. and Huber, W. (2010). EBImage—an R package for image processing with applications to cellular phenotypes. *Bioinformatics* **26**(7) 979–981.
[36] Picheny, V., Wagner, T. and Ginsbourger, D. (2013). A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization* **48**(3) 607–626.
[37] Prasad, A. and Alizadeh, E. (2019). Cell form and function: interpreting and controlling the shape of adherent cells. *Trends in Biotechnology* **37**(4) 347–357.
[38] Rasmussen, C. E. (2006) *Gaussian processes for Machine Learning*. MIT Press.
[39] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S. (2019). Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[40] Ridler, T. W., Calvard, S. et al. (1978). Picture thresholding using an iterative selection method. *IEEE Trans. Syst. Man Cybern* **8**(8) 630–632.
[41] Ronneberger, O., Fischer, P. and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18* 234–241. Springer.
[42] Roustant, O., Ginsbourger, D. and Deville, Y. (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software* **51**(1) 1–55. https://doi.org/10.18637/jss.v051.i01
[43] Rue, H., Martino, S. and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)* **71**(2) 319–392. https://doi.org/10.1111/j.1467-9868.2008.00700.x. MR2649602
[44] Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B. et al. (2012). Fiji: an open-source platform for biological-image analysis. *Nature Methods* **9**(7) 676–682.
[45] Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics* **656** 5–28. https://doi.org/10.1017/S0022112010001217. MR2669948
[46] Schneider, C. A., Rasband, W. S. and Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nature methods* **9**(7) 671–675.
[47] Selmeczi, D., Mosler, S., Hagedorn, P. H., Larsen, N. B. and Flyvbjerg, H. (2005). Cell motility as persistent random motion: theories from experiments. *Biophysical journal* **89**(2) 912–931.
[48] Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems* **18** 1257.
[49] Soetaert, K. E., Petzoldt, T. and Setzer, R. W. (2010). Solving differential equations in R: package deSolve. *Journal of Statistical Software* **33**(9).
[50] Stringer, C., Wang, T., Michaelos, M. and Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods* **18**(1) 100–106.
[51] Tinevez, J. -Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E., Bednarek, S. Y., Shorte, S. L. and Eliceiri, K. W. (2017). TrackMate: an open and extensible platform for single-particle tracking. *Methods* **115** 80–90.
[52] Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **61**(3) 611–622. https://doi.org/10.1111/1467-9868.00196. MR1707864

[53] Tsukui, T., Sun, K. -H., Wetter, J. B., Wilson-Kanamori, J. R., Hazelwood, L. A., Henderson, N. C., Adams, T. S., Schupp, J. C., Poli, S. D., Rosas, I. O. et al. (2020). Collagen-producing lung cell atlas identifies multiple subsets with distinct localization and relevance to fibrosis. *Nature communications* **11**(1) 1920.

[54] Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L. and Kutz, J. N. (2014). On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics* **1**(2) 391–421. https://doi.org/10.3934/jcd.2014.1.391. MR3415261

[55] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(6) 583–598. https://doi.org/10.1109/34.87344

[56] Wu, K. -J., Edmund, C., Shang, C. and Guo, Z. (2022). Nucleation and growth in solution synthesis of nanostructures–from fundamentals to advanced applications. *Progress in Materials Science* **123** 100821.

[57] Zhu, Y., Peruzzi, M., Li, C. and Dunson, D. B. (2024). Radial neighbours for provably accurate scalable approximations of Gaussian processes. *Biometrika* **111**(4) 1151–1167. https://doi.org/10.1093/biomet/asae029. MR4830051

Laura Baracaldo. Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. E-mail address: lnbaracaldol@ucsb.edu

Blythe King. Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. E-mail address: blytheking@umail.ucsb.edu

Haoran Yan. Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. E-mail address: haoranyan@umail.ucsb.edu

Yizi Lin. Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. E-mail address: lin768@umail.ucsb.edu

Nina Miolane. Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA. E-mail address: ninamiolane@ucsb.edu

Mengyang Gu. Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. E-mail address: mengyang@pstat.ucsb.edu