Biomedical Optics EXPRESS

Gaussian process post-processing for particle tracking velocimetry

TOMMY TANG,[1] ENGIN DENIZ,[2] MUSTAFA K. KHOKHA, [2,3] AND HEMANT D. TAGARE[1,4,*]

[1] *Department of Radiology and Biomedical Imaging, Yale University, 300 Cedar St, New Haven, CT 06510, USA*
[2] *Pediatric Genomics Discovery Program, Department of Pediatrics, Yale University, 333 Cedar Street, New Haven, CT 06510, USA*
[3] *Department of Genetics, Yale University, 333 Cedar Street, New Haven, CT 06510, USA*
[4] *Department of {Biomedical Engineering, Electrical Engineering, Statistics and Data Science}, Yale University, New Haven, CT 06510, USA*
[*] *hemant.tagare@yale.edu*

Abstract: Particle tracking velocimetry (PTV) gives quantitative estimates of fluid flow velocities from images. But particle tracking is a complicated problem, and it often produces results that need substantial post-processing. We propose a novel Gaussian process regression-based post-processing step for PTV: The method smooths ("denoises") and densely interpolates velocity estimates while rejecting track irregularities. The method works under a large range of particle densities and fluid velocities. In addition, the method calculates standard deviances (error bars) for the velocity estimates, opening the possibility of propagating the standard deviances through subsequent processing and analysis. The accuracy of the method is experimentally evaluated using Optical Coherence Tomography images of particles in laminar flow in a pipe phantom. Following this, the method is used to quantify cilia-driven fluid flow and vorticity patterns in a developing *Xenopus* embryo.

1. Introduction

Image-based particle tracking velocimetry (PTV) tracks freely floating particles in a fluid and uses the tracks to estimate the underlying fluid flow velocities [1–3]. The idea behind PTV is conceptually appealing, but in reality, particle tracking is a complicated affair. Many real-world phenomenon – such as the appearance and disappearance of particles – confuse the particle tracker, and corrupt its velocity estimates. As a result, PTV velocities need significant post-processing to be useful. This paper presents a Gaussian process regression (GPR) framework for such post-processing. Our method accepts noisy PTV output and produces improved fluid flow velocity estimates.

The proposed method works for sparse as well as dense particle concentrations, a wide spread of fluid velocities, and is also robust to tracking irregularities. The method offers another feature: it calculates the standard deviation of the velocity estimates, effectively giving "error bars" for the results. To the best of our knowledge, no other PTV or PTV post-processing method offers this. The calculated standard deviations can be used either to assess the reliability of velocity estimates, or they can be propagated further downstream to obtain error bars for subsequent calculations.

We apply our method to two sets of experiments. In both experiments Optical Coherence Tomography (OCT) is used to image particles flowing in the fluid. OCT creates an image sequence (a video) of a two-dimensional (2d) section through the three-dimensional (3d) fluid flow. We use our method to estimate the in-plane 2d fluid velocities from this image sequence.

The first set of experiments validate the method with a laminar-flow pipe phantom and assesses

#362207 https://doi.org/10.1364/BOE.10.003196
Journal © 2019 Received 12 Mar 2019; revised 14 May 2019; accepted 14 May 2019; published 7 Jun 2019

the method's performance. The second set of experiments applies the method to a biological problem: that of quantifying cilia-driven fluid flow in the embryo of the developing African clawed frog, *Xenopus*. *Xenopus* has provided multiple insights into human development and disease [3–5]. Similar to the human respiratory epithelium, the epidermis on the surface of the *Xenopus* embryo has an array of multiciliated cells that drive polarized flow from the head to the tail of the embryo [6]. These multiciliated cells can be easily assayed by OCT imaging [7]. Cilia movement and the cilia-driven fluid flow changes significantly during the development of the *Xenopus* embryo; this phenomenon and its relation to *Xenopus* genetic expression and to fluid mechanics is poorly understood. Quantifying cilia-driven fluid flow is a key step in understanding this relation. Quantifying cilia-driven fluid flow has broader implications as well: it is relevant to understanding respiratory disease, which is the primary cause of morbidity and mortality in the human pediatric population, accounting for 6.6 million deaths annually under the age of 5 [8].

Alternate techniques have also been reported in the literature for OCT fluid velocity estimation. These techniques include dynamic light scattering [9, 10], Doppler-OCT [11], particle streak velocimetry [12], and digital particle image velocimetry (DPIV) [9,13–15]. While PIV is a popular method for faster flows, in our preliminary investigations we found PIV to be overwhelmed by speckle in slower flows. Significant, data-dependent user interaction was necessary to counteract this effect.

Post-processing of tracks to improve velocity estimates has been reported in the literature [16–19]; most of these methods rely on adaptive Gaussian weighting (AGW) to smooth and interpolate the velocity fields. In spite of its popularity, AGW has limitations which are described in detail below in Section 2.1.

1.1. Organization of the paper

Section 2 contains an overview of our entire processing pipe-line: from OCT images to the final velocity estimates. The mathematics of GPR is discussed in detail in Section 2.4.4. Section 3 contains results of both sets of experiments, while Section 4 contains a discussion and Section 5 concludes the paper.

2. Methods

2.1. Motivation

To motivate the design of our algorithm, we begin by identifying real-world problems with PTV that a useful post-processing algorithm should take care of. An example from a real-world image sequence illustrating these problems is provided later on.

2.1.1. Track outliers

PTV tracks particles by assuming that the particle is present in consecutive image frames. However, images are a 2d section through a 3d flow, and particles are not persistent in 2d sections - old particles disappear as they move out of the imaging plane, and new particles appear as they enter into the imaging plane. In addition, images contain speckle which a tracker can mistake for a particle. All of this causes trackers to spuriously link a particle in one frame to the wrong particle or to speckle in the next frame. These tracking irregularities causes tracks to "jump" and give abnormal "outlier" fluid velocities. Post-processing is required to identify and remove these outlier velocities.

Our algorithm incorporates a method for outlier removal. The method is based on the assumption that fluid flow is stationary (not time varying) over the extent of the image sequence. Under this assumption, a track irregularity can be detected by comparing a track velocity with velocities of tracks in nearby image locations throughout the entire image sequence.

2.1.2. Track noise

Even when a track has no irregularities, the track can display significant frame-to-frame noise. This is largely due to noise in the image affecting particle localization, especially when trackers attempt sub-pixel localization. Using the frame-to-frame track displacements to calculate flow velocities gives very noisy velocity estimates; in slow velocity flows, the noise-to-signal ratio is particularly high and hinders meaningful velocity estimates. A simple solution to this is to calculate velocities by a linear fit to the tracks over multiple frames.

2.1.3. Particle density and interpolation

Particle density in the fluid has a significant effect on PTV. Because PTV only estimates velocities at the particle locations, when particles are sparse, the velocities have to be interpolated over long spatial gaps. Moreover, PTV velocities are noisy. Thus, post-processing is required to smooth velocities and simultaneously interpolate them.

To be useful, the velocity interpolation method should exhibit independent control over (1) the length of the gap it interpolates over, and (2) the closeness of fit to the measured data (if the measured data is low- or non-noisy, then the interpolation should be close to the measured data at the sites of the measured data). AGW does not allow independent control over interpolation length and closeness-of-fit. To understand why consider an informal example: Suppose we have low-noise measurements on a regular, but coarse, grid. In order to interpolate over the gaps in the grid, AGW must have a kernel larger than the grid spacing. This means that the interpolated values *at the original measurements* are a substantially smoothed version of the measurements, and thus quite different from the original low-noise measurements. In other words, AGW interpolation is biased. On the other hand, GPR gives completely unbiased estimates (under the observation model of Eq. (3) in section 2.4.4). This is because GPR has extra degrees of freedom, so that it can independently control interpolation length and closeness to data. And in addition, GPR offers closed form variance estimates for the velocities. These advantages motivated us to design a post-processing stage using GPR.

2.2. Simplifications

Our algorithm is designed for 2d images, i.e. for estimating the 2d in-plane component of the true 3d fluid velocities. The latter are highly constrained by the physics of fluid flow, e.g. 3d velocities are required to be tangential to the fluid boundary and the divergence of the velocities is required to be zero. However, these constraints only apply to the full 3d velocities. It is straightforward to show that a 2d projection of the 3d velocities need not satisfy these constraints. Consequently, we do not impose these constraints. This simplifies post-processing considerably.

2.3. Algorithm overview

Based on the above considerations, our algorithm processes PTV tracks in multiple stages. The entire processing pipeline – from the OCT images to the final velocity estimates – is illustrated in Fig. 1. The figure shows the input sequence of images as a 3d volume with time along one axis and the image plane along the other two.

The processing pipeline has three stages: The first stage uses an off-the-shelf tracker to obtain a set of tracks from the images. The first stage also segments the images to get the region containing the fluid (velocity is estimated only within the segmented region). The second stage smooths the track by fitting straight lines to the track over multiple frames and then exploits stationarity to reject outliers. This reduces the data to a set of 2d velocity vectors at discrete points in the segmented fluid region. The third stage smooths and interpolates these vectors using GPR to give a dense vector field in all pixels of the segmented region. Details of each of these stages are given below. The second and the third stages are the post-processing stages.

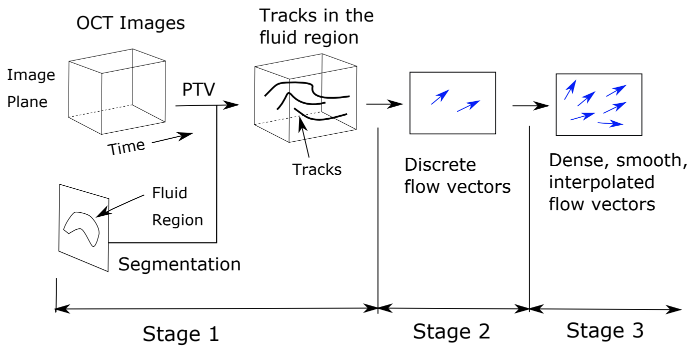


Fig. 1. The processing pipeline. Stage 1 consists of the PTV and the the fluid region segmenter. Stage 2 creates velocity vectors from tracks, projects them on the image, and deletes outliers. Stage 3 smoothes and interpolates the vectors to get a dense flow velocity map.

The motivation for splitting the algorithm into three stages is to enforce modularity – we want the flexibility to swap in and out different trackers and discretization algorithms without altering the GPR part of the code.

2.4. *Algorithm stages*

2.4.1. The first stage

As mentioned above, the first stage uses a particle tracking algorithm and a segmentation algorithm. The specific particle tracking algorithm we use is TrackMate, a Fiji (an enhanced version of ImageJ) plugin [20–22]. TrackMate produces a set of tracks as an output, where each track is a sequence of image locations and the corresponding time values. We use LAP tracking; alternative methods in TrackMate do not provide significantly different results.

The segmentation algorithm used to isolate the fluid region is interactive. The user initializes a seed in the fluid region, and a level set algorithm is used to grow the seed to the entire fluid region. The user can interactively correct segmentation results, if needed. Tracks whose image locations lie within the segmented region are retained. All other tracks are discarded. The retained tracks are the output of the first stage and are passed as input to the second stage.

Processing in subsequent stages is independent of the specific tracker and segmentation algorithm used in the first stage. All that is required of the first stage is that it produce a set of tracks in the fluid region. To test insensitivity to the tracker, we have also successfully used Mosaic [23] as the particle tracker in the first stage. However all results in this paper are obtained with TrackMate.

2.4.2. The second stage

In the second stage, the image sequence is partitioned into consecutive groups of N frames (by default $N = 33$), see Fig. 2(a). Within each group of frames, every track is approximated by a straight line segment. The approximation is carried out as weighted least-squares fit to the track, with the weight being a Gaussian located at the center of the group of frames. The center and the end of the best-fit line segment defines a displacement vector, which when divided by the time difference, gives the velocity vector of the track. We associate the start of the velocity vector with the center of the line segment, and then project the center and the velocity vector onto the image plane (see Fig. 2(a)). Line fitting and projection are standard signal processing methods, and to save space, we omit formulae for them.

After processing all tracks in this way, we have a set of locations (projected centers of the line

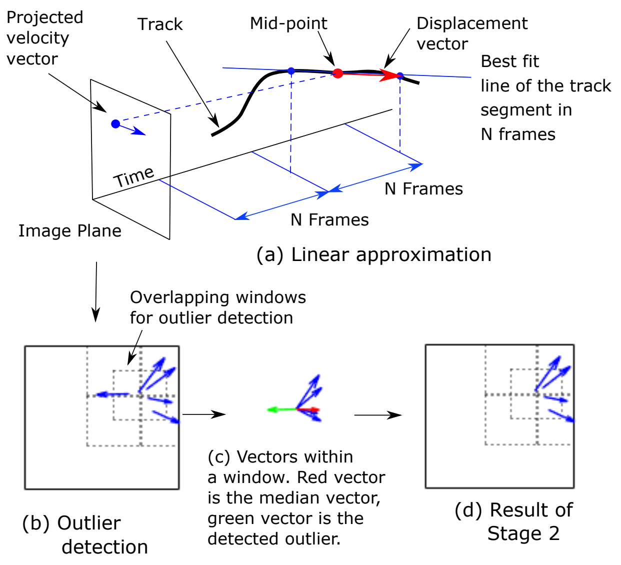

Projected velocity vector
Track
Mid-point
Displacement vector
Best fit line of the track segment in N frames
Time
N Frames
N Frames
Image Plane
(a) Linear approximation
Overlapping windows for outlier detection
(b) Outlier detection
(c) Vectors within a window. Red vector is the median vector, green vector is the detected outlier.
(d) Result of Stage 2

Fig. 2. Details of Stage 2 of the algorithm. (a) Image sequence is grouped into consecutive N frames. Each track in one set of set of frames is approximated by straight line, and the velocity vector calculated from this is projected onto the image plane. (b) Overlapping windows are used to reject outlier velocities. (c) Vectors within a window are shifted to a common origin and compared with the median vector to identify and delete outliers (see text for details), and (d) the output is a discrete set of noisy flow vectors in the image. Note that the vectors displayed above in (b),(c), and (d) are only illustrative, and are not the result of applying the algorithm to any data set.

segments) in the image plane and 2-d velocity vectors (projected velocities) at these locations. Together, the two are represented as $(\vec{v}_i, x_i)$, $i = 1, \cdots, k$, where the vector $\vec{v}_i$ is located at x_i. Since the flow is stationary and smooth, we expect velocity vectors in nearby locations to be similar. This observation provides a means of identifying and rejecting outliers: we tile the image plane with overlapping windows (as shown in Fig. 2(b)) and within each window we calculate the median velocity, rejecting as outliers those vectors that differ greatly from the median velocity. To be precise, if $(\vec{v}_i, x_i)$, $i = 1, \cdots, k_1$ are the vectors and locations in a window, then moving the tails of the vectors to a common origin (see Fig. 2(c)), we define the median vector $\vec{v}^* = (v_x^*, v_y^*)$ by

$$v_x^* = \text{median}(\vec{v}_{1,x}, \cdots, \vec{v}_{k_1,x}), \quad \text{and} \quad v_y^* = \text{median}(\vec{v}_{1,y}, \cdots, \vec{v}_{k_1,y}),$$

where the subscripts x, y refer to the x- and y-components of the vectors. Referring to Fig. 2(c), the median vector (depicted in red), is the “center” of the spray of vectors $\vec{v}_i$, $i = 1, \cdots, k_1$. If $d_i = \|\vec{v}_i - \vec{v}^*\|$ is the length of the difference vector between v_i and the median vector, then $D = \text{median}(d_1, \cdots, d_{k_1})$ is the median length of the difference vectors. All vectors $\vec{v}_i$ for which $d_i > 2 \times D$ are taken as outliers. Outlier vectors and their corresponding locations are deleted.

Note that because the windows overlap (Fig. 2(b)), every vector is contained in two windows. A vector is considered an outlier if it is an outlier in either of the two windows. Overlapping windows are used to avoid spurious results for vectors that are near the boundary of a window.

Thus, at the end of the second stage we have a collection of (noisy, but not outlier) velocity vectors located in the plane (Fig. 2(d)). We continue to denote this set as $(\vec{v}_i, x_i)$, $i = 1, \cdots, m$, where $\vec{v}_i$ is the vector at location x_i.

Figure 3 shows an example of Stage 2 input and output (the example is taken from the “high-flow” sequence described in Section 3). Tracks from Stage 1 are shown in blue. The frame-to-frame noise in the tracks is clearly visible, as is the "jump" from one particle to another in the right hand side of the figure. If we calculate velocity using the frame-to-frame displacements, then the resulting velocities will be quite noisy; this is clear from the jagged appearance of the tracks. The tracks in the figure suggest that fitting straight lines to the tracks over multiple frames and then calculating velocities from the fitted lines will ameliorate this problem. The arrows in Fig. 3 show the velocities found by the linear fitting from Stage 2 of the algorithm (the magnitude of the arrows has been scaled by 10 for display). The color of the arrows indicate whether Stage 2 found the velocity to be an inlier (green) or an outlier (red). Clearly, Stage 2 has not only obtained smoother velocity estimates but has also rejected the obvious outlier.

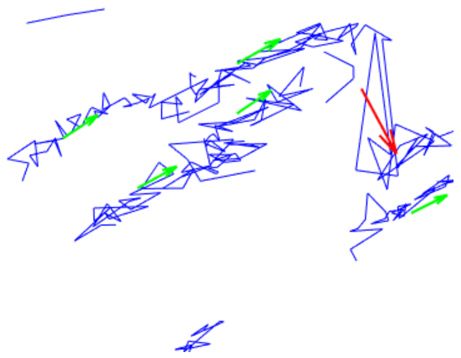


Fig. 3. A zoomed-in portion of the high-flow sequence, demonstrating the inputs and output of Stage 2. The blue curves are tracks found by TrackMate. The arrows are the velocities found by linear fitting in Stage 2, scaled by 10 for visibility. The red arrow is a velocity detected as an outlier and deleted, while the green arrows are inliers and used as input for Stage 3

Note that the idea of detecting outliers using the median of the deviation from the median is commonly used (for example, it is available in MATLAB as a generic outlier detector function isoutlier()). If needed, the outlier detection step can be made more sophisticated by finding the multivariate median of the vectors (rather than the component-wise median). This may be a more accurate calculation of the median, but we are not interested in the exact value of the median; we are only interested in rejecting obvious outliers. The component-wise median works well for that purpose, and is easy to implement.

2.4.3. The third stage

In the third stage, GPR is used to smooth and interpolate the velocities over all pixels of the fluid region. Because there are no constraints on the 2d velocity field, the $x-$ and $y-$ components of the velocity vectors are processed independently. We generically refer to the either of these components as v_i, without the over-arrow. Of course, v_i is a scalar, and the main concern in the third stage is to smoothly interpolate these scalar values. This is done via GPR.

2.4.4. Gaussian process regression

The idea behind GPR is to create a set of non-linear functions in such a way that specifying this set is simple, and fitting a specific function from this set to sparse data is computationally tractable. In GPR, the set of functions is taken to be sample functions of a stationary Gaussian stochastic process [24, 25].Thus GPR is completely specified by its mean and convariance functions. The covariance function is also called the *kernel function*. The mean of the stochastic process (the mean is a function) a-priori biases the set of functions to vary around the mean, while the kernel function of the stochastic process determines (a-priori) the smoothness of the functions. If no prior knowledge is available, then the set of functions need not be biased to vary around a predetermined mean, and the mean of the GPR is set to 0. The kernel function of the GPR is usually specified in a parametric form; its parameters are hyperparameters of the problem and can be determined from the data (see Section 2.5). Given noisy scalar values at some locations, the Gaussian process assumption gives the conditional probability density of the values at any other set of locations. The mean of the conditional probability density is the Bayes estimate of the values at these other locations. The standard deviation of the conditional probability density is the "error bar" of the value. The mean and the standard deviation are easily calculated from the conditional density. The calculations, which are available in the machine learning literature [24, 25], are specialized to our problem thus:

Let Ω be the set of all points in the segmented fluid region, and let μ be a Gaussian random process defined on Ω. Assume that μ has zero mean and a covariance (kernel) function

$$k(x_1, x_2) = \theta_s^2 \exp(-||x_1 - x_2||^2/2\theta_w^2) \tag{1}$$

where x_1, x_2 are points in Ω, and $\theta_s, \theta_w > 0$ are parameters that determine the width of the kernel function and the squared norm of the data. Because μ is a Gaussian process, if $x_1, \cdots, x_N$ are any points in Ω, the vector $(\mu(x_1), \cdots, \mu(x_N))^T$ has a Gaussian probability density $\mathscr{N}(0, K)$, where K is an $n \times n$ matrix with $K_{ij} = k(x_i, x_j)$. Further suppose that $(\mu(x_1), \cdots, \mu(x_N))^T$ is split into two groups:

$$\mu = \begin{pmatrix} \mu(x_1) \\ \vdots \\ \mu(x_m) \end{pmatrix}, \ \mu^* = \begin{pmatrix} \mu(x_{m+1}) \\ \vdots \\ \mu(x_N) \end{pmatrix}, \tag{2}$$

and that we have noisy observations of the first group, i.e. we observe

$$\nu = \mu + \epsilon, \tag{3}$$

where, ϵ is zero mean Gaussian noise with a diagonal covariance matrix $\sigma_n^2 I$. Given ν, we wish to estimate μ^*. The Bayes estimate of μ^* and its standard deviation is obtained via the conditional density of μ^* given ν. To be clear, this means that we calculate the conditional probability density $p(\mu^* \mid \nu)$ and take the mean of the conditional density to be the Bayes estimate of μ^*. The diagonal of the covariance matrix of the conditional density gives the variances, and hence the standard deviations, of the Bayes estimate.

To see the relevance of this formulation to our problem, recall that Stage 2 of our algorithm produces a set of velocity vectors and their locations. We take the $x-$ (and then, the $y-$) components of the velocity to be the noisy observations ν_i, $i = 1, \cdots, m$ in Eq. (3) and the locations to be x_i, $i = 1, \cdots, m$. In addition, we take the locations of all pixels in the segmented fluid region to be $x_i, i = m+1, \cdots, N$. Then, according to the above argument, the conditional density of μ^* given ν gives the Bayes estimates of the velocities and their standard deviations at the fluid region pixels.

To calculate the conditional density of μ^* given ν, we begin by noting that conditioned on μ, the random variables μ^* and ν are independent, and their conditional densities are Gaussian ($p(\mu^* \mid \mu)$ is Gaussian because μ is a Gaussian process, and $p(\nu \mid \mu)$ is Gaussian because our model for noise is Gaussian). Since the marginal density of μ is Gaussian (because μ is a Gaussian process), the joint density $p(\mu^*, \nu, \mu) = p(\mu^* \mid \mu)p(\nu \mid \mu)p(\mu)$ is also Gaussian. It is straightforward to show that the joint density has zero mean. Therefore, the marginal density $p(\mu^*, \nu)$ is also Gaussian with zero mean. The correlations between the components of μ^* and ν are:

$$\begin{aligned} E\left[\mu_i^*\mu_j^*\right] &= k(x_{m+i}, x_{m+j}), \\ E\left[\mu_i^*\nu_j\right] &= E\left[\mu_i^*(\mu_j+\epsilon_j)\right] = k(x_{m+i}, x_j), \\ E\left[\nu_i\nu_j\right] &= E\left[(\mu_i+\epsilon_i)(\mu_j+\epsilon_j)\right] = k(x_i,x_j)+\sigma_n^2\delta_{ij}, \end{aligned} \tag{4}$$

where k is the kernel function of Eq. (1), and δ_{ij} is the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$, else $\delta_{ij} = 0$).

Thus, the density $p(\mu^*, \nu) = \mathcal{N}(0, C)$, where

$$C = \begin{pmatrix} C_{\mu^*\mu^*} & C_{\mu^*\nu} \\ C_{\mu^*\nu}^T & C_{\nu\nu}. \end{pmatrix} \tag{5}$$

Here, $C_{\mu^*\mu^*}$ is an $N - m \times N - m$ matrix with $C_{\mu^*\mu^*}[i,j] = E\left[\mu_i^*\mu_j^*\right]$, $C_{\mu^*\nu}$ is an $N - m \times m$ matrix with $C_{\mu^*\nu}[i,j] = E\left[\mu_i^*\nu_j\right]$, and $C_{\nu\nu}$ is an $m \times m$ matrix with $C_{\nu\nu}[i,j] = E\left[\nu_i\nu_j\right]$. The conditional density $p(\mu^* \mid \nu)$ follows immediately:

$$p(\mu^* \mid \nu) = \mathcal{N}(C_{\mu^*\nu}C_{\nu\nu}^{-1}\nu, C_{\mu^*\mu^*} - C_{\mu^*\nu}C_{\nu\nu}^{-1}C_{\mu^*\nu}^T). \tag{6}$$

Therefore, the conditional mean of μ^* given ν is $C_{\mu^*\nu}C_{\nu\nu}^{-1}\nu$, which is the Bayes estimate of μ^* given ν. The conditional variance of each component of the μ^* is the corresponding term in the diagonal of $C_{\mu^*\mu^*} - C_{\mu^*\nu}C_{\nu\nu}^{-1}C_{\mu^*\nu}^T$. The square root of the conditional variance is the standard deviation.

To summarize: The input to Stage 3 is the set of vector,location pairs $(\overrightarrow{v}_i, x_i)$, $i = 1, \cdots, m$. The x- and y-components of the vectors are processed independently. The conditional mean and standard deviation of each velocity component is estimated over all pixels in the fluid region using Eq. (6). The estimates of the x- and y- components are conjoined to give the Bayes estimate of the velocity at every pixel in the fluid region.

2.5. Algorithm parameters

Recall that the covariance function of Eq. (1) has two hyperparameters: θ_s^2, θ_w^2. The noise variance σ_n^2 is an additional hyperparameters in the formulation above. We set these parameters as follows: θ_s^2, which is the variance of the zero-mean process μ is approximated by setting it to the mean squared norm of the two-dimensional velocities. After this, θ_w^2 and σ_n^2 are determined by maximizing the log-likelihood of data [25]:

$$(\theta_w^2, \sigma_n^2) = \underset{\theta_w^2,\sigma_n^2}{\arg\max} \log p(\nu_1, \cdots, \nu_m | \theta_w, \sigma_n^2). \tag{7}$$

Note that σ_n^2 and θ_w^2 are calculated separately for the x and y velocity components, while θ_s^2 takes the same value for both. The parameters θ_s^2, θ_w^2 and σ_n^2 are determined first from data, and then the conditional mean and variance of μ^* are calculated according to Eq. (6). The result is that GPR contains no free parameters. The only free parameter in the algorithm is N, which is used in Stage 2 and is set to $N = 33$, as mentioned before.

2.6. Visualization

The velocity estimates produced by the algorithm are visualized as arrows in the fluid region. In addition, we also visualize the magnitude of the velocity components, the standard deviation of the velocity estimates, and the vorticity of the velocity field. The magnitude is displayed as a color map of the square root of the sum of the squares of the x- and y-components. The standard deviation of the velocity estimate is calculated as the square root of the sum of the variances of the x- and y-components of the velocity at every pixel, and is also displayed as a color map. The vorticity is calculated as the ratio of the curl of the velocity field divided by the magnitude of the velocity field. Because the curl of any vector field linearly scales with the vector field, dividing by the magnitude enables a clear visualization of vorticity in the slow as well as fast moving parts of the fluid.

2.7. Implementation

As noted above, we use TrackMate for creating tracks in the first stage of the algorithm. The level set segmentation algorithm and the second and third stages of the algorithm are coded in MATLAB [26]. The third stage is implemented using the Gaussian Process Regression Toolbox available in MATLAB.

The parameters used for TrackMate are selected to eliminate obvious tracking errors. In particular, we select parameters to avoid identifying OCT speckle as particles. This is done by selecting a TrackMate quality threshold that eliminates approximately 15 percent of identified particles.

3. Results

We now describe the four sets of results. The first set validates the algorithm using laminar fluid flow in a cylindrical pipe. The theoretical velocity profile for such a flow is well-known, and serves as the "gold standard" with which to compare the algorithm velocity estimates. The second set of results uses the algorithm to estimate cilia-driven fluid flow velocities for a developing *Xenopus* embryo at different stages of maturity. In the third set of results, we tested self-consistency of our algorithm by splitting each Xenopus video into two subvideos, first into odd and even-numbered frames, and then into first and second halves and compared the difference in magnitudes to the norm of the model variance. In the fourth set of results, we explored the effect of varying the data density on the GPR estimate.

The images for all experiments were obtained with the Thorlabs Ganymede Series SD-OCT System (36kHz A-Scan Rate, Acquisition Time: 0.016 sec, Pixel size: 4μm x-axis / 1.31μm). Variable number of frames were acquired (number of frames is given below for each sequence).

The acquired video is difficult to display on paper. However if all frames in the sequence are summed together to obtain a single summed image, then flowing particles (which appear white against a dark fluid background) appear as streaks in the summed image (see Fig. 4(a), for example). Below, we display all acquired OCT image sequences as streak images. For easy visualization, the contrast in the displayed streak images is adjusted so that the streaks are easily seen. Of course, all algorithm results use the raw OCT images without any adjustment. For the Xenopus images, we also display a single frame in the sequence, so the reader may visually assess the relative appearance of speckle and particles.

All animal work was done in accordance with established guidelines and regulations and was approved by the Institutional Animal Care & Use Committee (IACUC) of the Yale School of Medicine.

We report TrackMate parameters used for each set of results in Data File 1. Although GPR hyperparameters are determined by the data through our algorithm, for sake of completeness, we report GPR hyperparameters obtained for each set of results in Data File 2.

3.1. Experiment 1: validation with a pipe phantom

A syringe pump (New Era Pump Systems) and polyethylene tubing (pipe) of 0.7mm inner diameter were connected to a 10ml syringe. PolyStyrene beads (Bangs Laboratories, mean diameter 0.96μm) were infused through the setup. The beads were dissolved in 1/9 Modified Ringers salt, and the bead-dissolved flow in the pipe was imaged under OCT. OCT imaging clearly displayed the flowing beads. The OCT imaging plane was set to contain the pipe long-axis in the horizontal direction with the flow directed from the right to the left. 336 frames of the OCT video were acquired.

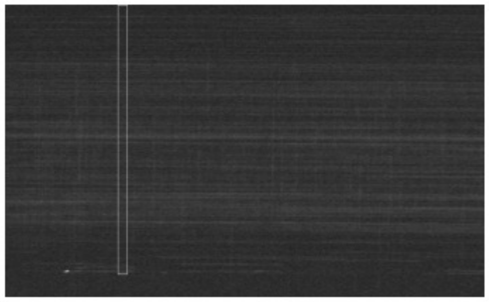
Fluid dynamics theory [27] shows that the flow velocities in the image should be horizontally directed, with a magnitude that is greatest at the center of the pipe and decreasing quadratically in the vertical direction. That is, we expect the x-component of the estimated flow velocity to change quadratically with respect to vertical distance in the image, while we expect the y-component of the velocity to be zero.

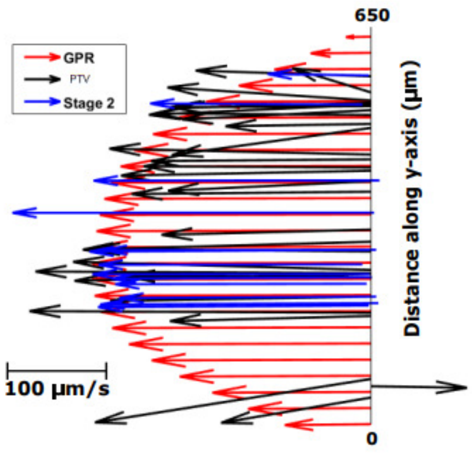
Figure 4(a) is the streak image for the pipe OCT images - the OCT video is given in Visualization 1. The horizontal flow of beads, and hence of the fluid, is visible in the streak image. Executing our algorithm with these OCT images gives a dense estimate of flow velocities in the image. To assess the efficacy of our method we compare the raw PTV velocities (PTV velocities directly calculated from the output of Stage 1 of the algorithm from frame-to-frame displacements), and the velocities at the output of Stage 2, with the GPR velocities. The three sets of velocities were gathered from a thin rectangle (20 pixels wide) shown in Fig. 4(a). The raw PTV velocities and the GPR output are quite dense in the rectangle, while Stage 2 velocities are much sparser (due to the piecewise linear fit). To visualize all of these meaningfully, in Fig. 4(b) we plot the raw PTV and GPR velocities only from the vertical mid-line of the rectangle, while we plot all of the Stage 2 velocities from the rectangle. It is clear from the figure that the PTV velocities (the black arrows in Fig. 4(b)) are noisy; in fact, the noise is strong enough to completely mask the quadratic velocity profile. The PTV velocities also show a tracking irregularity; a velocity near the bottom of the section points in the opposite direction of the flow. Very likely, this irregularity is the result of the tracker jumping from one particle to another particle or to speckle. The velocities at the output of Stage 2 (blue arrows) show that Stage 2 has removed the outlier velocity. However, the quadratic velocity profile is still not completely apparent. The GPR velocities have significantly less noise, and are available at every pixel. These velocities are horizontal, and they demonstrate a clear quadratic magnitude profile.

Figures 4(c)-(d) along with Table 1 give a more quantitative analysis of PTV, Stage 2, and GPR velocities from the rectangle. Figures 4(c)-(d) show respectively the x- and y-components of PTV, Stage 2, and GPR velocities as a function of distance along the section. The figures also contain a fit of a quadratic function to the data (velocity component as a quadratic function of the distance along the section). There are far too many data points in the rectangle to plot in Figs. 4(c)-(d), so we downsampled PTV and Stage 2 outputs by 20 and 200 times, respectively, for display. However, the quadratic fits are made to all data points. These figures clearly show the noisy nature of PTV velocities.

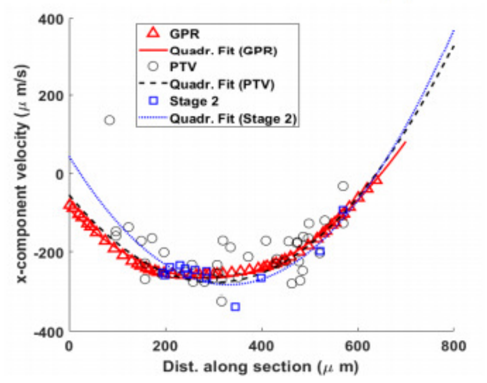
For each velocity component and method (PTV, Stage 2, or GPR) Table 1 shows the best fit quadratic polynomial to all data points in the rectangle, the mean square error of the fit, and the R value which indicates goodness-of-fit. By definition, R values are in the range $0 - 1$ with 1 representing a perfect fit to the data, and 0 representing “no fit” to the data (i.e the data is indistinguishable from pure noise). For the x-component, Table 1 shows that Stage 2 reduces the mean square error by about an order of magnitude, and then GPR reduces the mean square error by another order of magnitude. Thus, GPR has a mean square error that is two orders of magnitude less that that of PTV. Moreover, the R values show that the quadratic fit to GPR is almost perfect, while the fit to PTV is significantly noisy.

The results for the y-components are similar: Here too Stage 2 reduces the mean square error followed by further significant reduction by GPR. Recall that ideally the y-component of this horizontal flow should be zero. Figure 4(d) and the polynomial coefficients for GPR show that the y-component of GPR is very close to zero, and that there is very little noise in the component

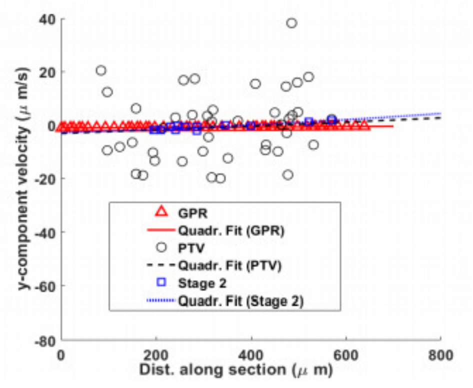

(a) Streak image for beads flowing through the pipe. The vertical rectangle is the section from which we obtain velocities for analysis. These velocities are visualized in (b)



(b) PTV, Stage 2, and GPR velocities along the section in the image



(c) x-component of the flow velocity as estimated by PTV, Stage 2, and GPR



(d) y-component of the flow velocity as estimated by PTV, Stage 2, and GPR

Fig. 4. PTV, Stage 2 and GPR velocity estimates for beads in a laminar flow in a cylindrical Pipe. (a) Streak image from the OCT video in Visualization 1, (b) PTV (black arrows), Stage 2 (blue arrows) and GPR velocity (red arrow) estimates, (c) - (d) the fit of a quadratic model to the x- and y- components of the velocity.

(R is 1.0 up to the first decimal).

3.2. *Experiment 2: cilia-driven fluid flow in the* Xenopus *embryo*

The *Xenopus* embryo develops in a sequence of well-defined stages, which are referred to numerically as stages 1 through 50 (http://www.xenbase.org). Stage 1 is the earliest stage, i.e. the freshly fertilized egg, and stage 50 is the tadpole stage. In the early stages, the *Xenopus* embryo develops in a sac, called the Vitelline membrane (see Fig. 6(b) for an illustration). Besides the embryo, the sac also contains fluid. When imaged with OCT, the fluid reveals floating particles. Tracking these particles quantifies cilia-driven fluid motion. Imaging reveals that there is no

Table 1. Quadratic fits to x- and y-components of PTV and GPR velocities. The variable d in the best fit polynomial is distance along the section. The R value indicates the goodness-of-fit; R values are between 0 and 1, with 1 being a perfect fit.

Velocity component	Method	Best fit polynomial (d = dist. along section)	Mean sq. err.	R
x	PTV	$0.0025d^2 - 1.48d - 54.88$	6398	0.28
	Stage 2	$0.003d^2 - 1.98d_43.65$	737.09	0.79
	GPR	$0.0021d^2 - 1.21d - 91.07$	39.8	0.99
y	PTV	$1.19 \times 10^{-6}d^2 + 0.006d - 3.12$	1005	0.001
	Stage 2	$4.28 \times 10^{-6}d^2 + 0.0055d - 2.83$	0.405	0.77
	GPR	$4.19 \times 10^{-7}d^2 + 6.30 \times 10^{-4}d - 1.08$	4.36×10^{-5}	1.0

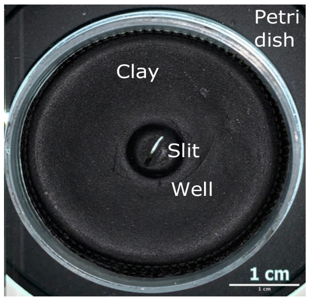


Fig. 5. Imaging preparation. Petri dish coated with clay in which a well is formed. *Xenopus* embryo are fixed in a slit in the well.

flow in the sac till developmental stages 19 – 21, after which flow initiates. Fully developed flow is established by stages 24 – 27. Particle density in the fluid region also changes with the developmental stage. Very few particles are present in the early developmental stages, the number growing substantially by stages 24 – 27.

We imaged the embryo with OCT begining at stage 16 and ending in stage 25 at hourly intervals. The Vitelline membrane remained intact throughout the imaging process. To prepare for imaging (Fig. 5), we coated the bottom of the petri dish (Polystyrene petri dish (35 × 10mm)) with clay (Wax based non-hardening clay (Van Aken Claytoon)) and formed a well (0.5cm diameter) at the center. Stage 16 *Xenopus* embryos were positioned at the center of the well in a slit, and remained in place throughout imaging.

The imaging plane was aligned over the long-axis of the embryo from the anterior to the posterior with the head of the embryo appearing in the right half of the image (Fig. 69b)). The dorsal side of the embryo appeared above the ventral side in all images. The Vitelline membrane was also visible in the images, with the fluid-filled region appearing between the Vitelline membrane and the dorsal side of the embryo. It is worth noting that during stages 16 – 24, a number of grooves appear on the dorsal side of the embryo due to the development of the pharyngeal arches. These grooves change the boundary conditions of the fluid flow.

All OCT video sequences were successfully processed with our algorithm. Below, we report results from three representative imaging sessions, approximately at developmental stages 19, 22, and 25. At these stages, the fluid flow is, respectively, barely established, has moderate velocity, and is fully established. We call these conditions *low-flow*, *mid-flow*, and *high-flow*. The number of OCT frames acquired in these flow conditions were 500, 918, and 918 respectively. The videos for low-, mid-, and high-flow are available as Visualization 2, Visualization 3, and Visualization 4 respectively.

Figure 6 shows the low-flow case. Figure 6(a) shows a single frame in the video Visualization 2. Figure 6(b) shows the streak image, with identifying labels. Figure 6(c) shows the velocity estimates produced by TrackMate in the fluid region. Note the sparse and noisy nature of the estimates. The estimates are sparse because there are very few particles in the fluid at this stage of the development. Figure 6(d) shows the output of our algorithm - the velocity estimates have been "denoised" and smoothly interpolated throughout the fluid region. Figures 6(e)-(g) show colormaps of the magnitude of the estimated velocity, the standard deviation of the velocity estimates, and the vorticity. Note the higher standard deviation of the velocity estimates near the top boundary of the extreme right corner of the fluid region. The higher standard deviation is due to lack of particles in this region, as apparent in the TrackMate output (Fig. 6(c)).

Figures 7-8 show similar results for the mid- and high-flow stages. These results are discussed further in the next Section.

3.3. Validation of *Xenopus* flow tracking

We tested the self-consistency of the algorithm with the Xenopus data, we split each Xenopus video into two subvideos, the first video containing only odd-numbered frames and the second video containing only even-numbered frames. We refer to these as the odd and even subvideos. Using TrackMate, we obtained tracks for the two subvideos and performed post-processing with our algorithm. The fluid region segmentation of the original video was used as the segmentation for both subvideos.

For every pixel in the segmentation, writing the estimated velocity from the odd subvideo as $\mu_1 = (\mu_{x,1}, \mu_{y,1})$ with variances $(\sigma^2_{x,1}, \sigma^2_{y,1})$ and the output from even subvideo as $\mu_2 = (\mu_{x,2}, \mu_{y,2})$ with variances $(\sigma^2_{x,2}, \sigma^2_{y,2})$, we computed the normalized difference

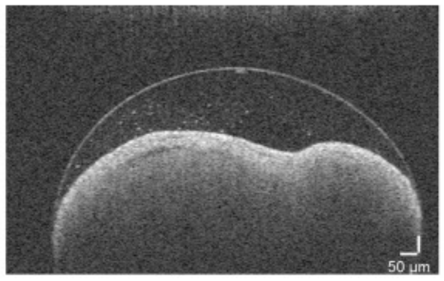
$$\frac{||\mu_1 - \mu_2||}{\sqrt{\sigma^2_{x,1} + \sigma^2_{y,1} + \sigma^2_{x,2} + \sigma^2_{y,2}}}. \tag{8}$$

If our algorithm is consistent, then most (but not all) of the normalized differences should be less than 1. Figure 9 a-c show the histograms of the normalized differences for all pixels in the segmented regions for low-, mid-, and high- flow. To be more quantitive, we calculated the fraction of pixels in the segmented region whose normalized differences are below 1. This fraction is reported in the second column of Table 2.

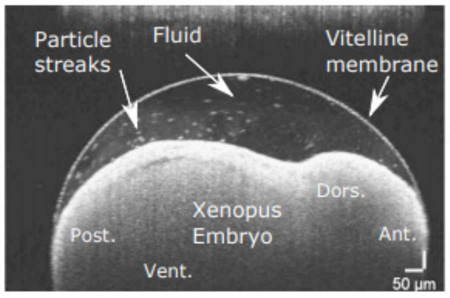
Table 2. Fraction of pixels in the segmented region whose normalized difference is less than 1

Data set	Odd/Even Subvideo	First/Second Half Subvideo
Xenopus Low-flow	0.99	0.92
Xenopus Med-flow	0.99	0.89
Xenopus High-flow	0.95	0.87

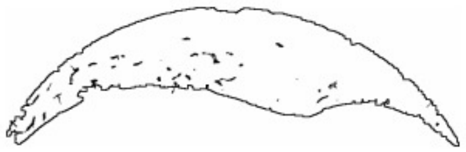
Next we evaluated the stationarity of flows by again splitting each Xenopus videos into two videos, the first containing the first half of the frames and the second containing the second half of the frames. As above, TrackMate was used to obtain tracks in each video, and the tracks were



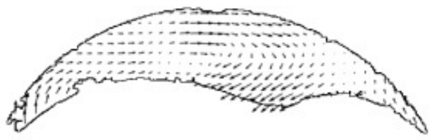
(a) Single Frame



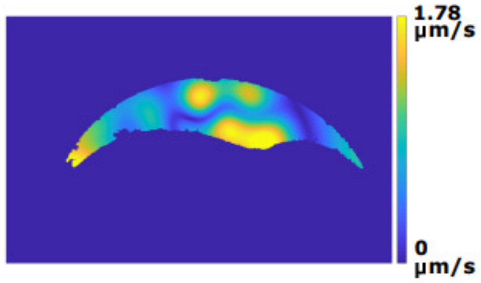
(b) Streak image



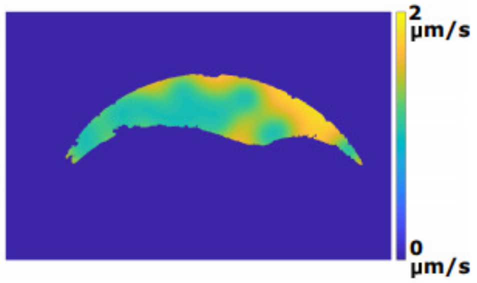
(c) TrackMate output velocities



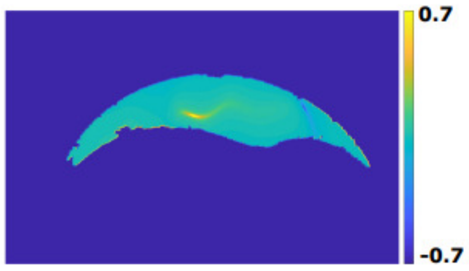
(d) Algorithm output velocities



(e) Velocity magnitude (μm/sec)



(f) Velocity standard deviation (μm/sec)



(g) Vorticity

Fig. 6. Fluid velocity estimates in the low-flow stage. Visualization 2 contains the OCT video, with a single frame shown in (a).

post-processed by our algorithm. The fluid region segmentation of the original video was used as the segmentation for both subvideos.The normalized difference was calculated as above. Figure

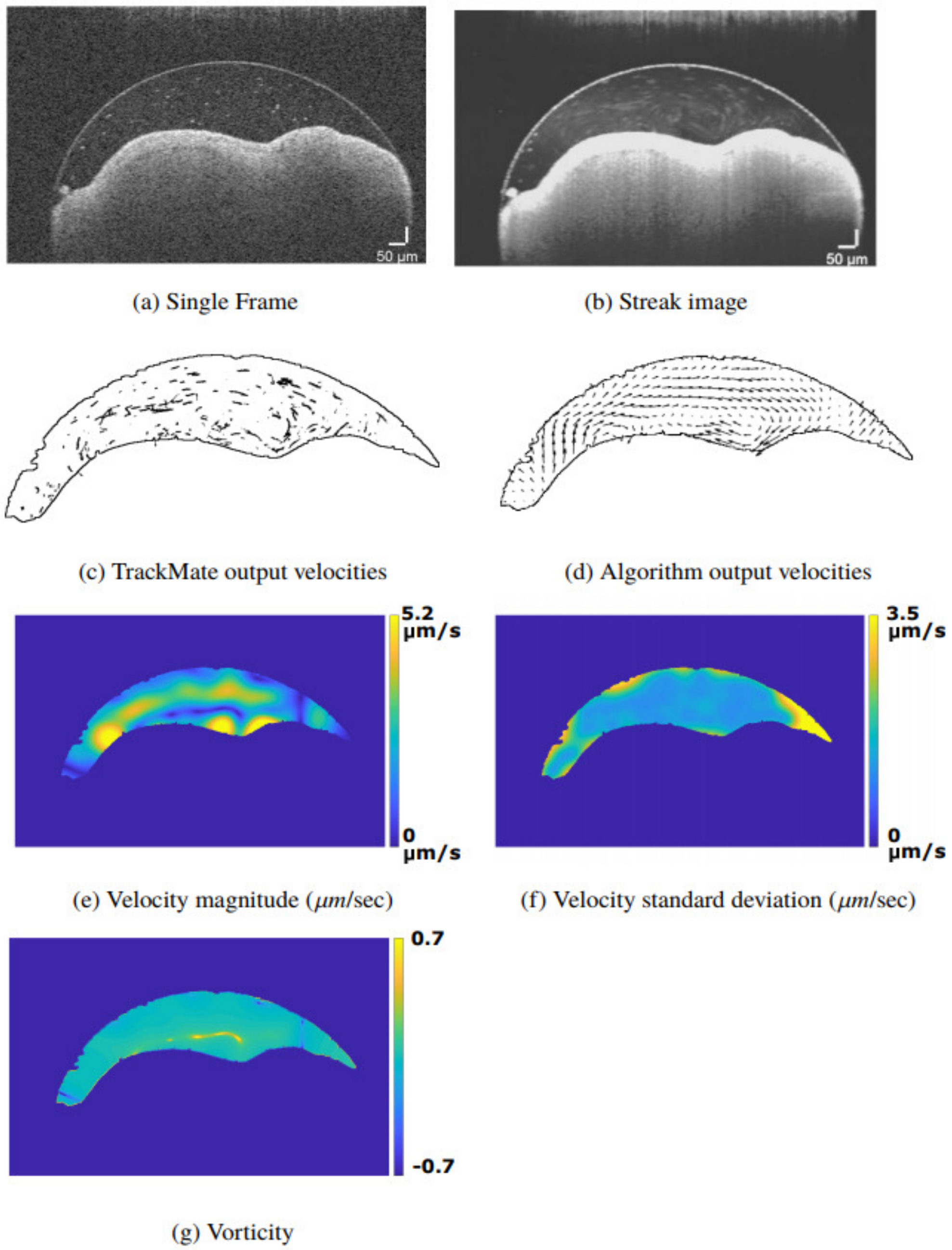
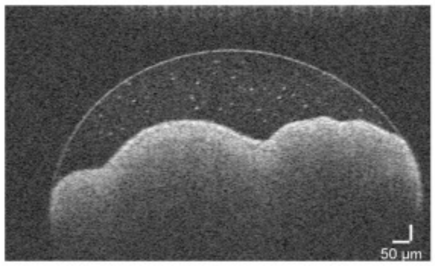
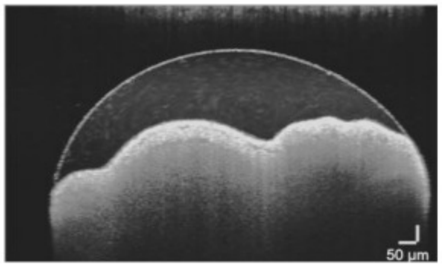

50 μm
50 μm
(a) Single Frame
(b) Streak image
(c) TrackMate output velocities
(d) Algorithm output velocities
5.2 μm/s
0 μm/s
3.5 μm/s
0 μm/s
(e) Velocity magnitude (μm/sec)
(f) Velocity standard deviation (μm/sec)
0.7
-0.7
(g) Vorticity

Fig. 7. Fluid velocity estimates in the mid-flow stage. Visualization 3 contains the OCT video, with a single frame shown in (a).

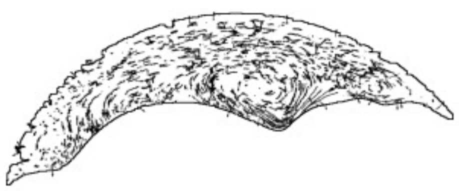
9 shows the histograms of the normalized differences, and the third column of Table 2 shows the fraction of pixels in the segmented region whose normalized differences are below 1.



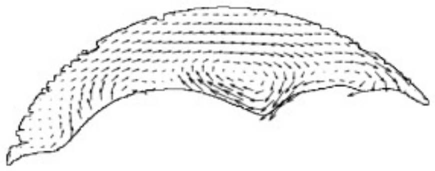
(a) Single Frame



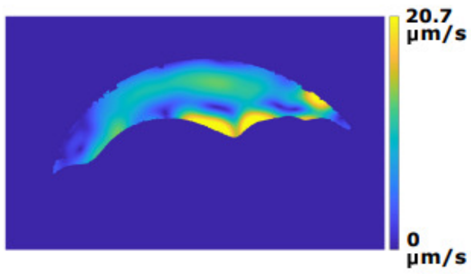
(b) Streak image



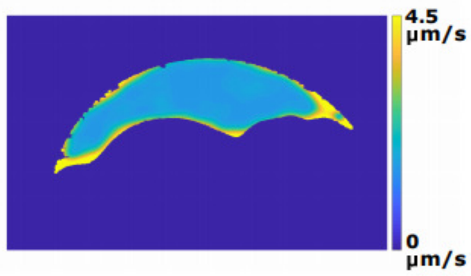
(c) TrackMate output velocities



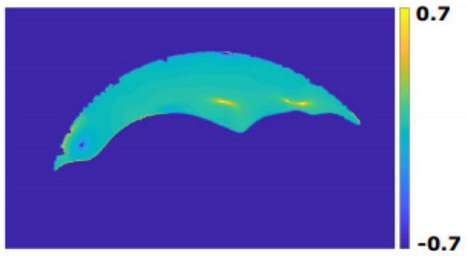
(d) Algorithm output velocities



(e) Velocity magnitude (μm/sec)



(f) Velocity standard deviation (μm/sec)

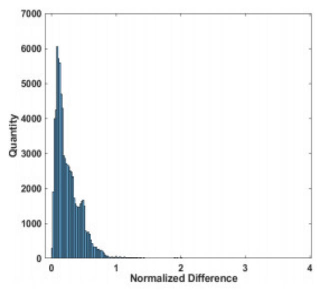


(g) Vorticity

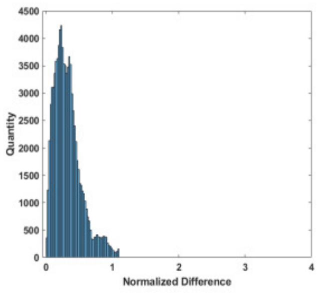
Fig. 8. Fluid velocity estimates in the high-flow stage. Visualization 4 contains the OCT video, with a single frame shown in (a).

3.4. *Response to variable data density*

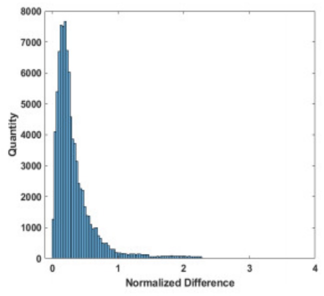
Finally, we turn to investigating the response of GPR to variable data density. GPR theory suggests that if data density decreases within a compact region in the image, then GPR should be able to successfully interpolate velocities from outside the region but the variance of this estimate



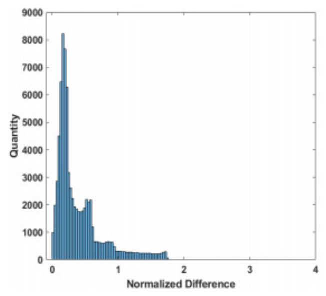
(a) low-flow evens/odds



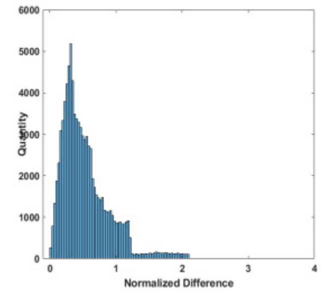
(b) mid-flow evens/odds



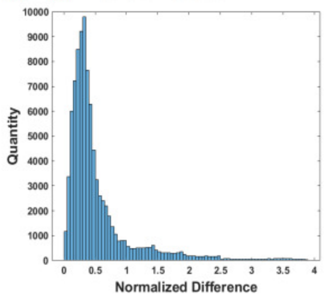
(c) high-flow evens/odds



(d) low-flow first and second half



(e) mid-flow first and second half



(f) high-flow first and second half

Fig. 9. Histograms of normalized differences of velocity estimates of subvideos

is likely to increase. To test this, we took the Stage 2 output for the high-flow *Xenopus* data and systematically deleted a certain fraction of Stage 2 output velocities from a rectangular region of size 230 × 120 pixels (approx. 300 × 160 μm). There were 245 velocity vectors in the region, from which we first deleted at random 50% vectors, and then deleted all (100%) vectors. The undeleted and deleted Stage 2 output are shown in the top row of Fig. 10. The rectangular region is indicated in the figure as well. Stage 3 (GPR) of our algorithm was used to find the entire flow field using these data. The middle row of Fig. 10 shows the estimated velocities and the bottom row of Fig. 10 shows the standard deviations.

Qualitatively, the middle and bottom rows of Fig. 10 show that GPR works as expected – the interpolated velocities within the rectangle are consistent, and more importantly, the standard deviation of the estimate within the rectangle increases with increasing deletion. The latter is particularly apparent in the 100% deletion case. For a more quantitative analysis we calculated the normalized difference (Eq. (8)) in the rectangular region between the 0% deleted velocities and 50% and 100% deleted velocities respectively. For 50% deletion, all normalized differences in the rectangular region were less than 1, while for 100% deletion, 88% of the normalized differences in the rectangular region were less than 1. Finally we measured the median standard deviation in the rectangular region for the undeleted, 50% deletion, and 100% deletion cases. These were 1.733 $\mu m/s$, 1.836 $\mu m/s$, and 5.646 $\mu m/s$ respectively, showing increased standard deviation with deletion.

4. Discussion

Velocity estimates produced by the algorithm with the pipe phantom data and the *Xenopus* embryo data clearly demonstrate the power of the algorithm to interpolate over a variety of

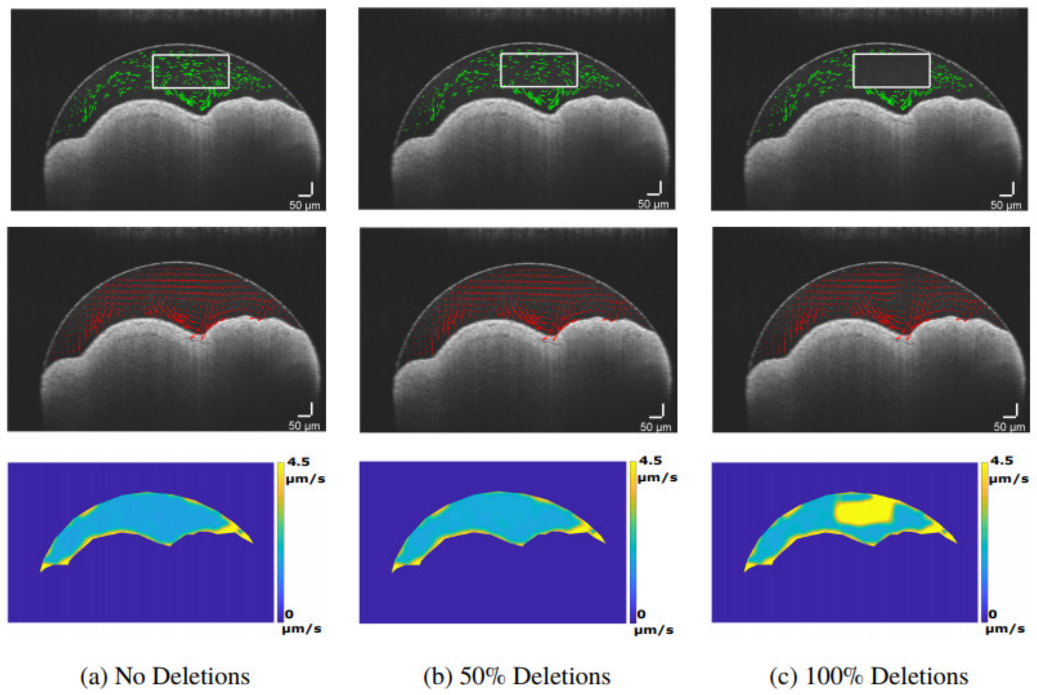


Fig. 10. GPR output for different levels of deletion in the rectangular region shown in the top row. (a) Left column: 0% deletion. (b) Middle column: 50% deletion. (c) Right column: 100% deletion. Top row: input to GPR. Middle row: GPR interpolated velocity estimates. Bottom row: Standard deviation of the velocity estimates.

particle densities and fluid velocities, to smooth the velocity field while rejecting outliers, and to calculate meaningful standard deviations for the results.

4.1. Particle densities and fluid velocities

The algorithm works over a large range of particle densities and fluid velocities. To demonstrate this quantitatively, we roughly estimated particle density in the image sequence as the total number of starting points of tracks found by the tracker divided by the area of the fluid region. Table 3 shows the particle densities and average GPR-estimated fluid velocities for the pipe-phantom and the three *Xenopus* embryo stages. Note that particle densities vary by a factor of 5 over the data sets, and that flow velocities vary by more than a factor of 100 over the data sets. No manual adjustments are needed to the algorithm to operate over this range.

4.2. Smoothing and outlier rejection

The algorithm smoothes the velocity estimates while rejecting outliers. This is best illustrated by the results of the pipe phantom where the exact theoretical velocity profile is known. As shown in Table 1, using GPR reduces the mean square error by almost two orders of magnitude, and gives close to perfect fits to the theoretical profiles. The ability to reject outliers is also evident in Fig. 4(b), where the outlier velocity pointing to the right is successfully rejected.

Table 3. Track densities and average velocities of the data sets

Data set	Particle Density (μm^{-2})	Avg. Velocity (μm/sec)
Pipe	0.07	157.66 $\mu m/s$
Xenopus Low-flow	0.11	0.85 $\mu m/s$
Xenopus Med-flow	0.18	2.47 $\mu m/s$
Xenopus High-flow	0.39	8.63 $\mu m/s$

4.3. Velocity standard deviations

Figures 6-8 clearly show the utility of calculating velocity standard deviations estimates. High velocity standard deviations are clearly present in the “f” parts of the figure in the subregions where the particle density is low. Velocity estimates in these subregions are interpolated from tracks that are farther away, and hence the estimates are trusted to a lesser degree. This phenomenon is especially evident in the rightmost corner of the fluid regions in Figs. 6(f)-8(f) where the standard deviation decreases as the particle density increases.

The standard deviation of the estimate can be meaningfully propagated further in the computation pipeline. For example, the standard deviation can be used to calculated error bars in the estimate of the net kinetic energy of the fluid.

4.4. Vorticity

Because GPR smooths velocity estimates, the vorticity estimates are relatively noise free and easy to interpret. Vorticity estimates in the *Xenopus* data sets are particularly intriguing. The vorticity of the flow appears to move towards the anterior as the embryo develops from low-flow stage to a high flow-stage. Furthermore, a second focused vortex seems to develop in the high-flow stage close to the newly formed grove near the anterior end of the embryo (compare Figs. 7(g) and 8(g)), suggesting a pairing of the vortices with the groves. We hope to explore the relation between flow vorticity and *Xenopus* embryo development more thoroughly in the near future. For now, this preliminary finding clearly demonstrates the advantage of using smooth, GPR post-processed, velocities in quantifying flow.

4.5. Consistency and stationarity

The histograms of the odd and even subvideos in Fig. 9(a)-(c) and the entries in column 2 of Table 2 shows that the algorithm is self-consistent. The vast majority of pixels in fluid region have normalized differences less than 1. Similarly, the histograms of the first and second half subvideos in Fig. 9(d)-(f) and the entries in column 3 of Table 2 show that flow is stationary.

4.6. Variable data density

Figure 10 shows that the Bayesian estimates from GPR work as expected when the data density changes. With 50% data deletion, GPR uses the remaining data to give estimates that are consistent with no deletion, albeit with increased standard deviation. With 100% data deletion, GPR interpolates to give consistent estimates, but indicates that the standard deviation of the estimate is significantly higher.

5. Conclusion

Traditional signal processing methods conjoined with GPR gives a powerful post-processing algorithm for particle tracking velocimetry. The algorithm works without manual tweaking over a large range of particle densities and fluid velocities; it also provides substantial noise reduction and outlier rejection. When used to quantify cilia-driven fluid flow, the algorithm reveals flow vorticities and their relation to developmental stages.

Although we report the results of using the algorithm with OCT data, the algorithm can be used in other particle tracking applications as well. The GPR part of the algorithm is generic and can replace the velocity interpolation module of any other 2D flow estimator.

We plan to use this algorithm to disentangle and explore the genetic basis of cilia development and flow organization in the Xenopus embryo and also to disentangle the relative contributions of cilia to the cerebro-spinal fluid flow in the brain of the developed Xenopus embryo. Another avenue of research is to develop a fully 3D version of this algorithm that accounts for some of the physics of fluid flow.

Funding

This work was supported by the National Institutes of Health grants 5R01HL11849 (MKK, HT) and 5R33HL120783 (MKK). MKK is a Mallinckrodt Scholar.

Acknowledgments

The authors thank Sarah Kubek and Michael Slocum for animal husbandry.

Disclosures

The authors declare that there are no conflicts of interest related to this article.

References

1. S. Jonas, D. Bhattacharya, M. K. Khokha, and M. A. Choma, "Microfluidic characterization of cilia-driven fluid flow using optical coherence tomography-based particle tracking velocimetry," Biomed. Opt. Express **2**, 2022–2034 (2011).
2. F. Pereira, H. Stüer, E. C. Graff, and M. Gharib, "Two-frame 3d particle tracking," Meas. Sci. Technol. **17**, 1680 (2006).
3. K. C. Zhou, B. K. Huang, H. Tagare, and M. A. Choma, "Improved velocimetry in optical coherence tomography using bayesian analysis," Biomed. Opt. Express **6**, 4796–4811 (2015).
4. A. S. Warkman and P. A. Krieg, "Xenopus as a model system for vertebrate heart development," in *Seminars in cell & developmental biology,* vol. 18 (Elsevier, 2007), pp. 46–53.
5. M. K. Khokha, "Xenopus white papers and resources: folding functional genomics and genetics into the frog," Genesis **50**, 133–142 (2012).
6. M. Werner and B. Mitchell, "Understanding ciliated epithelia: the power of xenopus," Genesis **50**, 176–185 (2012).
7. B. K. Huang, U. A. Gamm, S. Jonas, M. K. Khokha, and M. A. Choma, "Quantitative optical coherence tomography imaging of intermediate flow defect phenotypes in ciliary physiology and pathophysiology," J. Biomed. Opt. **20**, 030502 (2015).
8. H. J. Zar and T. W. Ferkol, "The global burden of respiratory disease - impact on child health," Pediatr. Pulmonol. **49**, 430–434 (2014).
9. B. K. Huang, U. A. Gamm, V. Bhandari, M. K. Khokha, and M. A. Choma, "Three-dimensional, three-vector-component velocimetry of cilia-driven fluid flow using correlation-based approaches in optical coherence tomography," Biomed. Opt. Express **6**, 3515–3538 (2015).
10. J. Lee, W. Wu, J. Y. Jiang, B. Zhu, and D. A. Boas, "Dynamic light scattering optical coherence tomography," Opt. Express **20**, 22262–22277 (2012).
11. W. Drexler, M. Liu, A. Kumar, T. Kamali, A. Unterhuber, and R. A. Leitgeb, "Optical coherence tomography today: speed, contrast, and multimodality," J. Biomed. Opt. **19**, 071412 (2014).
12. K. C. Zhou, B. K. Huang, U. A. Gamm, V. Bhandari, M. K. Khokha, and M. A. Choma, "Particle streak velocimetry-optical coherence tomography: a novel method for multidimensional imaging of microscale fluid flows," Biomed. Opt. Express **7**, 1590–1603 (2016).
13. W. Thielicke and E. Stamhuis, "Pivlab–towards user-friendly, affordable and accurate digital particle image velocimetry in matlab," J. Open Res. Softw. **2** (2014).
14. W. Thielicke, "The flapping flight of birds: Analysis and application," Ph.D. thesis, University of Groningen (2014).
15. W. Thielicke and E. J. Stamhuis, "Pivlab - time-resolved digital particle image velocimetry tool for matlab," (2018).
16. J. C. Agui and J. J., "On the performance of particle tracking," J. Fluid Mech. **185**, 447–468 (1987).

17. H. Stuer and S. Blaser, "Interpolation of scattered 3d ptv data to a rectangular grid," J. Fluid Mech. **185**, 447–468 (1987).
18. A. Vlasenko, E. C. C. Steele, and W. A. M. Nimmo-Smith, "A physics-enabled flow restoration algorithm for sparse piv and ptv measurements," Meas. Sci. Technol. **26**, 065301 (2015).
19. J.-T. Kim, D. Kim, A. Liberzon, and L. P. Chamorro, "Three-dimensional particle tracking velocimetry for turbulence applications: Case of a jet flow," J. Vis. Exp. **108**, e53745 (2016).
20. J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, "Fiji: an open-source platform for biological-image analysis," Nat. Methods **9**, 676 (2012).
21. J. Schindelin, C. T. Rueden, M. C. Hiner, and K. W. Eliceiri, "The imagej ecosystem: an open platform for biomedical image analysis," Mol. Rep. Dev. **82**, 518–529 (2015).
22. J.-Y. Tinevez, N. Perry, J. Schindelin, G. M. Hoopes, G. D. Reynolds, E. Laplantine, S. Y. Bednarek, S. L. Shorte, and K. W. Eliceiri, "Trackmate: An open and extensible platform for single-particle tracking," Methods **115**, 80–90 (2017).
23. I. F. Sbalzarini and P. Koumoutsakos, "Feature point tracking and trajectory analysis for video imaging in cell biology," J. Struct. Biol. **151**, 182–195 (2005).
24. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).
25. C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer Science+Business Media, LLC, New York City, 2006), 1st ed.
26. MATLAB, *version 9.4.0 (R2018a)* (The MathWorks Inc., Natick, Massachusetts, 2018).
27. J. M. Cimbala and Y. A. Cengel, *Essentials of Fluid Mechanics, Fundamentals and Applcations* (McGraw-Hill, 2008).