13th International Symposium on Particle Image Velocimetry – ISPIV 2019
Munich, Germany, July 22-24, 2019

An open-source Shake-the-Box method and its performance evaluation

Shiyong Tan[1], Ashwanth Salibindla[1], Ashik Ullah Mohammad Masuk[1], Rui Ni[1*]

[1] Johns Hopkins University, Department of mechanical engineering, Baltimore, USA

[*] corresponding author: rui.ni@jhu.edu

Abstract

An open-source Shake-the-Box (STB) was developed to benefit both the PIV and PTV communities by establishing a platform for algorithm development and uncertainty analysis of the advanced particle tracking method. The algorithm was based on the paper by Schanz et al. (2016) but with a different code architecture and revised subroutines. In this article, tests of several key steps in the algorithm will be discussed. This code has also been parallelized and optimized to be run on high-performance computing (HPC) clusters to enable processing multiple datasets at the same time, and the data processing can be an order of magnitude faster depending on the number of cores and jobs that can be occupied. To test the code, Johns Hopkins turbulence database for homogeneous isotropic turbulence were used to make a synthetic dataset, and this dataset was sampled at different image densities to evaluate the algorithm performance. The results show that the open-source STB is able to detect up to 99.9% of particles and recover the correct flow characteristics at the particle image density of 0.05 ppp. At the same particle image density, our code seems to run slightly faster than the reported value. Four metrics were designed to quantify the quality of particle trajectories, and the results are almost perfect at least for the synthetic data. In addition to the synthetic data, the open-source STB was also tested on an experiment. The calibration, including the in-house calibration method and volume self-calibration, is discussed in details. A brief demonstration of the final results shows that this open-source code is ready to be used for experiments.

1 Introduction

In experimental fluid dynamics, particle-based velocity measurements rely on imaging the motion of many small tracer particles that faithfully follow the flow. These images can be analyzed either by using the particle image correlation, i.e. Particle Image Velocimetry (PIV, (Adrian, 1984)), or tracking individual particles, i.e. Particle Tracking Velocimetry (PTV, (Papantoniou and Dracos, 1989; Nishino et al., 1989)). The particle image density is one of the most important parameters of both systems because it determines the spatial resolution and quality of the results. The particle image density on image is typically determined by a quantity called particle per pixel (ppp). For example, 100 particles on an image made of 1000×1000 pixels results in a particle image density of 10^{-4} ppp. Particle tracking method started from the early work of Chiu and Rib (1956) to obtain 3D position of particle by using only two cameras. To avoid any ambiguity, the image density had to be kept really low at 10^{-5} ppp (Sheu et al., 1982; Chang et al., 1984; Adamczyk and Rimai, 1988).

Maas et al. (1993) addressed this problem by using three or four cameras, and the particle image density increases to 10^{-4} ppp. Malik et al. (1993) also proposed to connect particles frame by frame into tracks by minimizing the change of acceleration. This algorithm was improved by Ouellette et al. (2006) using a method called the Four-frame Best Estimate (4BE), and the particle image density of 0.005 ppp was achieved. This method has been used in many systems, including convection (Ni et al., 2011), porous medium flow (Shen and Ni, 2017), and animal collective motion (Ni et al., 2015b).

To resolve the velocity gradient that requires a high spatial resolution, both Hoyer et al. (2005) and Ni et al. (2015a) developed the scanning particle tracking method. In these two papers, the particle image density reached more than 0.005 ppp but at a price. The temporal resolution of the camera was traded for a higher particle image density as the entire view volume is not illuminated at once but scanned over several frames using a laser slab. Since similar number of particles can be tracked in each slab, the total particle image density is higher. But it requires a camera with a much higher frame rate.

In comparison with PTV, the particle image density is much higher in 3D Tomographic PIV (Tomo-PIV), which can reach as high as 0.1 ppp (Kähler et al., 2016) because: (i) 3D Tomo-PIV only reconstructs the particle intensity field, not isolated particles; and (ii) the reconstruction allows and produces ghost particles, even though it is not preferred. One of the effective reconstruction methods is called multiplicative algebraic reconstruction technique (MART) (Elsinga et al., 2006). An important progress has been made by Wieneke (2012) to switch the reconstruction from the intensity field to the particle field by using the iterative particle reconstruction (IPR). There are two innovative steps in IPR that helps reconstruct a dense particle field: (i) shaking: iteratively refine particle 3D position and check the reprojection with particle images, and (ii) intensity check: use intensity information to remove ghost particles.

IPR only deals with individual frames. Shake-the-Box (STB) method (Schanz et al., 2016) uses and extends the principle of IPR to the time domain, and STB also enforces the temporal coherence of trajectories in particle reconstruction. These implementations were revolutionary, and STB becomes the state-of-the-art for time-resolved 3D velocity measurements. After its first paper, it has been successfully applied to acquire different types of flow statistics such as velocity (Schneiders and Scarano (2016)), vorticity and dissipation (Schneiders et al. (2017)), pressure field (Van Gent et al. (2017)), as well as the coherent structure (Schlueter-Kuck and Dabiri (2017)). It was also adapted and improved by Novara et al. (2015) to a multi-pulse system, which can be used for high-speed flow. Novara et al. (2016b) also modified STB to solve the issue of ghost particles at the initial phase of STB by using two independent imaging systems.

STB has many benefits, and the most important one is the significantly-reduced processing time compared with Tomo-PIV at a similar particle image density. It was reported in (Schanz et al. (2016)) that the total processing time of STB for 500 images with around 12,800 particles is around 69 min for one pass and 63 min for the second pass, which has greatly reduced the processing time from 650 h to just 2 h. Despite the great improvement of the algorithm, the time-resolved measurements generate large datasets, and the total processing time of multiple datasets that can be easily generated within seconds is still overwhelmingly large if running on a desktop. Running data analysis on a high-performance computing (HPC) system or a Graphics Processing Unit (GPU) based system becomes an important step.

In order to provide an easy and open access to the STB algorithm and allow processing large amount of data, an open-source STB code has been developed and will be provided on the GitHub repository (@JHU-NI-LAB) once finalized. The general principle of the open-source STB and some improvements on STB are presented in Section 2. In Section 3, the performance of the open-source STB and its comparison with the results reported in Schanz et al. (2016) are carefully examined by using the synthetic data. The open-source STB was also tested with our experimental data. The details of the setup will be given in Section 4. Section 5 will briefly discuss the results of an experiment analyzed by the open-source STB.

2 The open-source STB

The goal of this paper is to introduce an open-source STB code that is designed as a platform to improve the method and compare different algorithms. In addition, users can easily access the intermediate steps to conduct the uncertainty analysis. The code was written in C++, an object-based language that is robust and versatile on different platforms. The code was originally written in the Windows system and later transferred to the Linux system. This code has recently been optimized for running on high-performance computing (HPC) clusters because of the growing needs for analyzing large datasets. Once STB is implemented on HPC, the image preprocessing and data postprocessing have also been shifted to HPC so the entire processing time can be reduced for more than one order of magnitude. In this section, we will briefly introduce the structure of the open-source STB code. Different choices and improvements that we have made during the development of open-source STB will also be discussed.

2.1 The structure of the open-source STB code

The STB code input image sequences of all cameras and camera calibration files. The calibration files include camera parameters and optical-transfer functions (OTF), which are acquired separately in MATLAB codes. These calibration parameters are input into the STB code via two basic classes named *Camera* and *OTF*, respectively.

In the open-source STB, based on the hierarchy, the highest level of classes that form the backbone of the code is level I. There are three level-I classes that correspond to three major components of the code:

particle reconstruction (*Calibration*), IPR (*IPR*), and STB (*STB*).

Both *Calibration* and *STB* share four level-II classes to handle image loading (*Tiffload*), 2D particle identification (*Frame*), and 3D particle positions (*Position*), and position refinement (*Shaking*). For each frame, images from all four cameras will go through the level-I class *Calibration* for image pre-processing.

In addition to image pre-processing, *calibration* contains three subroutines, stereomatching, triangulation, and match pruning, all contained in a function named *Stereomatch*. During stereomatching, 2D points are matched with several possible combinations to find the 3D positions, but not all of these combinations are real. Match pruning only picks the combinations with the minimum triangulation error, and also optimizes the combinations to make sure that every 2D point can only be used in one combination. After match pruning, particle positions are refined in level-II class *Shaking*. *Shaking* serves two purposes: (i) refine the particle 3D positions, and (ii) remove possible ghost particles that fail the intensity check.

After *calibration*, the level-I class *IPR* implements the algorithm of iterative particle reconstruction. This class is a loop, iteratively conducting particle reconstruction and shaking by using class *Calibration* mentioned above.

The third level-I class *STB* is the main component of STB. Within *STB*, there are two functions, *InitialPhase* and *ConvergedPhase*. *InitialPhase* only deals with the first four frames of the dataset because these four frames do not have velocity field to make predictions. *InitialPhase* first calls the class *IPR* to identify particles. Particles from the first four frames are fed to a class called *PredictiveField* (level-II), in which particle-space correlation is conducted to connect points into tracks. After this, *ConvergedPhase* takes the velocity field obtained from *InitialPhase* to continue the tracks to the next frame. The residual particles that cannot be connected to the existing tracks will be processed through IPR to initialize new tracks. A basic class called *Track* is designed to manage track data, including make decisions of terminating a bad track. Finally, the input and output are handled by an interface class called *DataIO*.

During the development of the open-source STB, we have encountered several challenges. Here, we introduce some of the critical steps of STB that requires attention and further investigations.

2.2 Wiener filter

Since STB relies on extrapolating an existing track to obtain the particle location in the next frame, the algorithm of extrapolation becomes important. Schanz et al. (2016) introduced the Wiener filter method and we would like to compare the performance of this filter with a simple polynomial filter to see how it affects the trajectory length and quality.

Wiener filter (Proakis and Manolakis, 2007) is a linear invariant filter in the form of

$$x(n+1) = \sum_{i=0}^{N-1} a_i s(n-i) \tag{1}$$

where $s(n)$ is the observed noisy process containing the signal, $x(n)$ is the output at time n, and N is the order. The parameters of Wiener filter are calculated by minimizing the mean square error (MSE) as follows:

$$a_i = \arg\min E(e^2(n)) \tag{2}$$

where $e(n) = x(n) - s(n)$. The known vector $S_{n-1} = [s_{n-N}, s_{n-N+1}, ..., s_{n-1}]$ is used to obtain the parameters $A_N = [a_1, a_2, ..., a_N]$ of Wiener filter by recursively updating the parameters through:

$$A_N = A_N + \mu S_{n-1}^T \cdot (s_n - A_N \cdot S_{n-1}) \tag{3}$$

where μ is the step length which is usually taken as $1/(S_{n-1}^T \cdot S_{n-1})$, until the estimated error $e(n) = s_n - A_N \cdot S_{n-1}$ reaches a specific precision. The next point is then predicted through equation 1 by using the parameters A_N. In the open-source STB, based on several trials, 5^{th}-order Wiener filter is adopted to obtain the optimal prediction.

Polynomials filter uses the polynomials

$$x(n) = \sum_{i=0}^{N} b_i n^i \tag{4}$$

to fit data points by minimizing the MSE in the same form as that in equation 2.

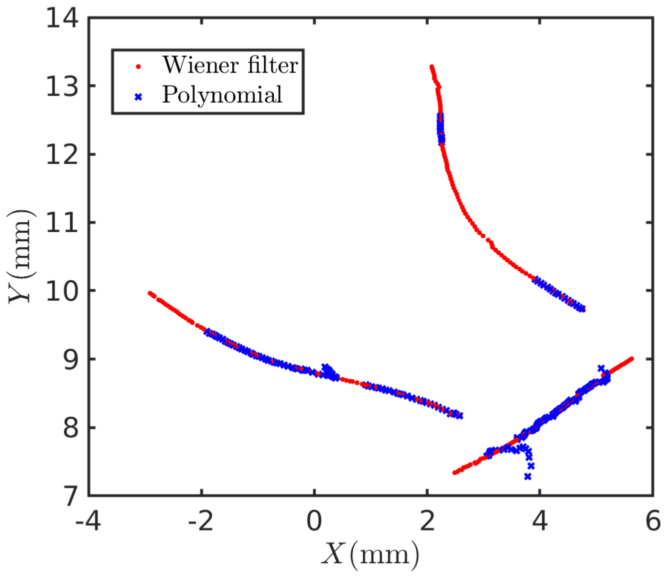

Figure 1: Sample trajectories identified by either the Wiener filter (red) or polynomials (blue) from a dataset at the particle image density of 0.0125 ppp.

For an extremely low image density (<0.005 ppp), STB based on both the Wiener filter and polynomials produces similar results with no discernible difference in either the number or the quality of trajectories. As shown in figure 1, at the particle image density of 0.0125 ppp, two problems for polynomials start to develop: (i) the trajectories become shorter and more broken, and (ii) two or more short particle tracks get incorrectly connected. We tried polynomials with different degrees from 3 to 10, and these two problems persisted. In comparison, the Wiener filter is more robust and produce high-quality particle tracks even at a high image density case.

The key difference between the two filters resides in their sensitivity to noise. At a high particle image density, overlapping particles lead to enhanced position noise and ambiguity. Extrapolating from a noisy trajectory could result in a large prediction error if the noise has not been handled properly. Since the Wiener filter has proven performance in dealing with a random process, it shows a much better result in our test, which is consistent with the findings by Schanz et al. (2016).

2.3 Linear-fit check

When either the particle image density or the image noise is high, the probability of finding a wrong track or a ghost track starts to increase. A wrong track is essentially several segments of good tracks incorrectly connected into one, and a ghost track is the track made of ghost particles. Typically the ghost particle cannot be connected for many frames, but this may occur if the image noise is large enough. Linear-fit check is the simplest frame-by-frame examination of the trajectory quality, which is based on the assumption that the temporal resolution is so high that every four consecutive frames can be connected by a linear track with very small physical acceleration. If the measured acceleration turns out to be abnormally large, the linear-fit check will identify that. In practice, at frame number n, particles from four frames ($n-3$ to n) will be fitted with a linear function, and the difference between the last point of the track $X(n)$ and the linear-fitted result $\tilde{X}(n)$ is used as an indicator $e_f = X(n) - \tilde{X}(n)$. If e_f is larger than a threshold, the trajectory will be terminated and no longer be used for prediction, and a new track will start.

Figure 2 shows the PDF of good and bad tracks using an experimental dataset. For this dataset, no linear-fit check was used and the trajectory quality was evaluated after STB. The good tracks are smooth, and the bad ones are jagged because of the triangulation and shaking uncertainties. Once the tracks are categorized based on their smoothness, the PDF of e_f of two groups are shown. It is clear that the bad tracks, either wrong or ghost tracks, tend to have a larger linear-fit error e_f, compared with the good ones. e_f =0.018 mm (dashed line) seems to cleanly divide the tracks into two groups. This number is close to the calibration error, which is 0.014 mm.

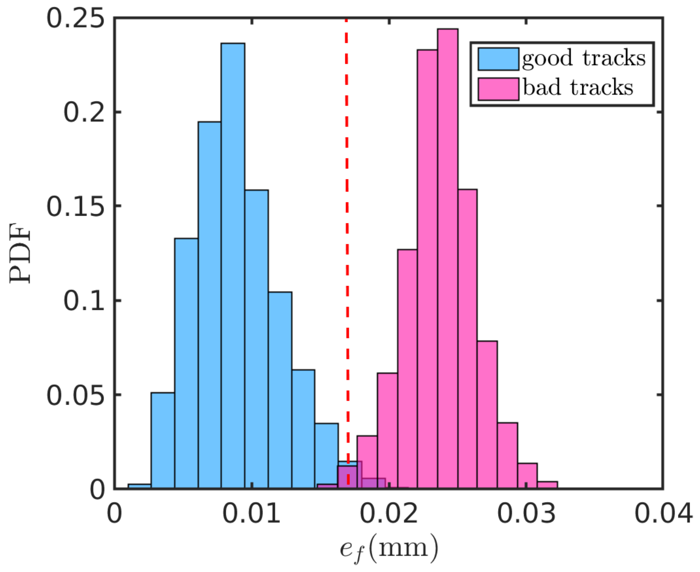

Figure 2: PDF of the linear-fit error e_f for good and bad tracks

In Schanz et al. (2016), the linear-fit check only examines the last data point and decides the fate of this track based on if e_f is above or below the threshold. This works well for removing most of the wrong tracks, but we found a new type of wrong track that cannot be effectively identified using this method. Those wrong tracks oscillate slightly within the threshold. Although this may be specific to experiments with large image noise in our back-lit particle tracking system, it may occur in other experimental configurations as well. To address this issue, for each track, the mean of the fitting error of all frames from 1 to *n* is calculated and used to examine the track. For the oscillating bad track, although all the fitting errors are slightly smaller than the threshold, the mean is usually more than 60% of the threshold.

Combining both the one-point *n* and multiple-point (1–*n*) checks provides a new advantage. We can set a large threshold in one-point check to allow the majority of the good tracks and some wrong ones to pass. This helps to make sure that the true high-acceleration events as part of the turbulence intermittency can be kept. Since the intermittent events will not last, the good tracks with these events will unlikely produce as large mean fit errors as the bad tracks will. As the result, the wrong tracks that pass the one-point check will be removed in the multiple-point check.

3 Parallelization

Although STB is much more computationally efficient than the tomographic reconstruction, its calculation is still the bottleneck of the entire experimental procedure as acquiring the high-speed video only takes seconds, transferring it only takes minutes, but STB will take hours to days depending on the number of frames, number of particles, calibration accuracy, and image noise level. Since STB involves many iterations in IPR and shaking, it is important to parallelize the code so that its performance in both the multi-processor desktop and high-performance computing cluster can be optimized.

As listed in table 1, three steps are computationally expensive, stereomatching, particle-space correlation, and shaking. Each subroutine has to iterate over all the particles. Stereomatching and shaking also have to iterate over all the camera images for each particle. To simplify the discussion, we assume the number of camera is a constant of four.

If the total number of particles in each camera image is roughly the same *N*, the stereomatching step is to find match of particle images from all four cameras to generate 3D particle positions. For each camera, the extrusion of each particle center along the optical axis of the camera into the 3D space becomes an epipolar line, and this line can be projected onto the imaging planes of all other cameras. Within certain distance (proportional to the calibration error) away from the epipolar line projection, *M* particles will be found, and these are possible candidates. The complexity of the calculation grows roughly with M^4, as each of these candidates has to be projected again onto other cameras to confirm the match.

The key step of STB is to establish the predictive field by using the particle-space correlation (Novara et al., 2016a), a step that identifies the velocity field based on particle displacement between two frames $n-1$ and n. As the displacement calculation involves correlating each particle in frame $n-1$ with another one in n, the calculation complexity grows with N^2.

The step of shaking can also be parallelized because it consists of several iterations for 3D position refinement and projections. To optimize the 3D particle location, they are iteratively "shaken" in 3D, starting from a large distance and gradually reduce to a smaller one. During each procedure, the updated 3D position will be projected onto all cameras to check with the actual particle images. The projection requires a 2D Gaussian calculation to represent the particle intensity profile. This calculation also scales with the total number of pixels that one particle occupies. In total, multiplying all the contributing factors results in a complexity estimation of $432N$ for particles of image diameter in 3 pixels.

OpenMP was adopted in our code to achieve the parallelization. It is not the most efficient method, but it is easy to implement. It evenly assigns the number of loops onto every available thread, thereby reducing the processing time. A example of the total processing time for each step with one or six cores of the same desktop is listed in table 1. The dataset used for tests is a synthetic data with 12,500 particles at a particle image density of 0.0125 ppp. A significant improvement of speed from parallelization can be noted in the table. For a higher particle image density, the time saved will increase.

Table 1: Processing time of stereomatching, particle-space correlation and shaking with parallelization

	Stereomatching	Particle-space correlation	Shaking
Computation	$N \cdot M^4$	N^2	$432 \cdot N$
One core (one thread used) (s)	46	180	45
Six cores (two threads per core) (s)	14	20	7

The open-source STB can be run on different platforms, including HPC clusters. For example, the Maryland Advanced Research Computing Center(MARCC) is a cluster consisted of 23,000 cores with 2.6 GHz frequency. Not only that the code can process one dataset using multiple processors, many datasets can also be processed at the same time by submitting several jobs together through the system, which can further decrease the processing time.

4 Performance evaluation

A performance test was conducted to evaluate different aspects of the open-source STB. Special attention has been paid to the uncertainty of the results and the efficiency of the algorithm.

4.1 Synthetic images

Synthetic images were generated by using an isotropic turbulence data from Johns Hopkins Turbulence database (JHTDB) (Li et al., 2008). Particles were tracked in the Lagrangian framework and their positions in each frame are projected onto four cameras with the image resolution 1024 × 1024 pixels. The particle intensity profile is assumed to follow a Gaussian function $I^i_{part}(x_i, y_i, p) = ae^{-(bx'^2+cy'^2)}$ with $x' = (x_i - x_{ip})\cos\alpha + (y_i - y_{ip})\sin\alpha$ and $y' = -(x_i - x_{ip})\sin\alpha + (y_i - y_{ip})\cos\alpha$. (x_i, y_i) is the pixel position and (x_{ip}, y_{ip}) is the center of the particle i. Constant $a = 255$, $b = 0.9$, $c = 0.9$ and $\alpha = 0$ were used to simplify the particle intensity profile, and these parameters can be calibrated using the optical transfer function (Schanz et al., 2012) that is designed to specify the projection parameters in each sub-volume. The average diameter of particles is around 3 pixels. Three sets of synthetic images with image density of 0.0125, 0.025, and 0.05 ppp were created.

4.2 Evaluation metrics

To evaluate the quality of the tracking results, each track identified by STB will be matched with a synthetic track. Every particle on the matched track was compared with each other. Four parameters were calculated to evaluate the tracking results:

Coverage: For each track, coverage is the ratio between the number of frames that can be linked by STB and the ideal track length from the synthetic data.

Matching error: The matching error is the separation between detected and synthetic particles. It is related to both the calibration error and triangulation error.

Fragmentation index: This index measures the brokenness of a track. If one complete track in the synthetic data becomes two broken ones after STB, this number equals to two.

Correctness: This number measures the percentage of real (not ghost) particles. Ghost particles refer to particles that do not actually exist but appear during the reconstruction. A number smaller than one indicates the existence of ghost particles.

4.3 Evaluation results

Only the parameter *coverage* was discussed by Schanz et al. (2016). For this particular parameter, the coverage reaches above 0.9991 even for images with image density of 0.05 ppp. Only 0.9% of particles are lost, which is slightly better than the 1.4% reported in Schanz et al. (2016). This may be attributed to the improved linear fit that retains more correct tracks. The matching error is about 0.001 mm for all image densities, which is much smaller than the average particle displacement of 0.06 mm between two consecutive frames. The fragmentation index for all image densities is around 1.03, which is very close to the ideal case of 1, suggesting that STB can detect almost all tracks without any break in the ideal zero-noise synthetic datasets. Correctness is also 1 for all three cases, which is again expected for a perfect dataset.

The algorithm was tested with a simple desktop with Intel i7-8700 six-core 3.2 GHz processors. For images with 12,500 particles, the processing time per frame is around 5 s. In Schanz et al. (2016), the code was run on a high-end server at that time with dual Intel Xeon E5-2680 ten-core processors at 2.8 GHz. The number of particles is 12,800, which is also similar to ours. The run time of our code is 40% faster than the one reported in Schanz et al. (2016). The number of cores used in our case is smaller, but each with a higher processing frequency. Therefore this direct comparison is not conclusive by any means because there are many things that are not controlled. But this at least suggests the performance of our code is comparable to, if not better than, the STB method reported by Schanz et al. (2016).

The code has also been run on MARCC with ten jobs, and the processing time per frame reduces to 1.2 s. Since MARCC is relatively old cluster, the computing power of each core is significantly slower than one core in a desktop but many cores can be used to speed up the process.

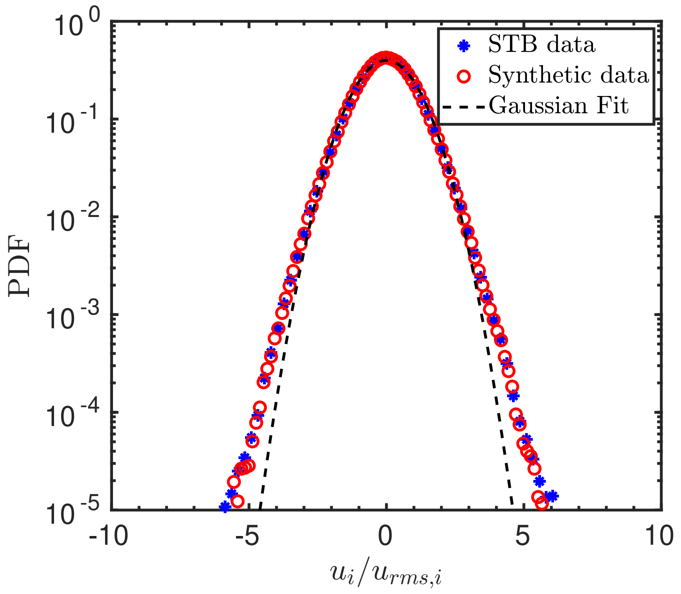

Figure 3: PDF of velocity fluctuation of both the synthetic data and the STB results at the particle image density of 0.05 ppp

The final test of comparing the probability density function (PDF) of the velocity fluctuation between the STB results and the simulation dataset with 0.05 ppp particle image density is shown in figure 3. It is clear that the blue symbols matches with the synthetic data very well, which shows that the open-source STB can accurately capture the flow characteristics.

5 Experimental setup and procedure

5.1 Experimental setup

The open-source STB was applied to a dataset from a vertical water tunnel with an octagonal test section. This tunnel was built to generate isotropic turbulence at a high energy dissipation rate. All eight faces of the test section were made of transparent acrylic plates. Four cameras were arranged around the octagonal test section with the top view shown in figure 4, and a picture of the test section with three cameras is also shown.

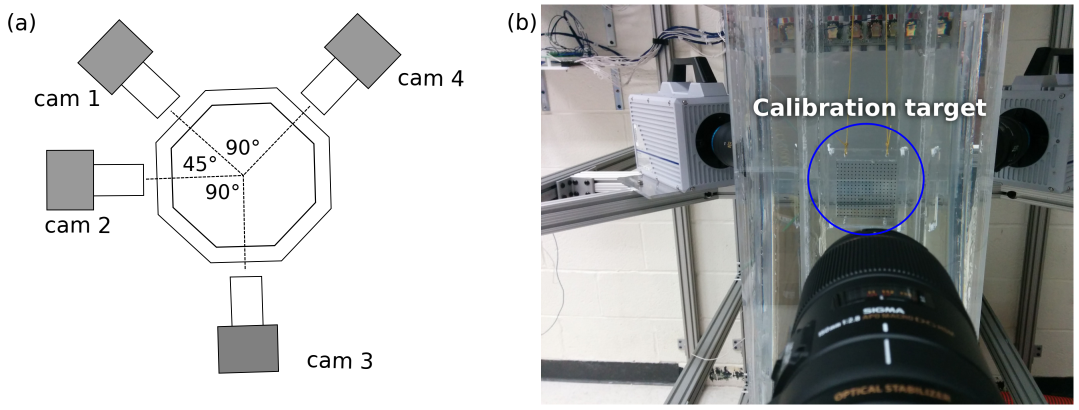

Figure 4: (a)Schematic of the camera positions and relative angles along with the octagonal test section, (b) Picture of a transparent calibration target in the setup with cameras and octagonl test section

5.2 Experimental procedure

Many factors tend to affect the performance of STB, such as lighting intensity, image noise, camera calibration, frame rate, particle size, and so on. Among those factors, the calibration step is key, which requires special attention. In order to obtain accurate camera parameters for STB, an experimental procedure was designed as follows:

1. Calibrate all cameras simultaneously with a (transparent) calibration target
2. Improve the calibration results using the Volume Self-calibration (VSC) on a dataset with a low image density of particles. Our VSC is different with the method introduced by Wieneke (2008).
   i. Seed flow with low-concentration of tracers so that the image density of particles on all cameras is lower than 0.001 ppp.
   ii. Stir the flow and record the data; run STB on the data with a large search radius to find all the possible tracks. Keeping a low image density helps to make sure that all identified tracks are real, not ghost.
   iii. Use the tracks that are longer than certain length to perform VSC
3. Conduct another VSC on high-image-density datasets.
   i. Directly conduct the actual experiment with the desired seeding density
   ii. Use a few frames to run STB and find tracks
   iii. Conduct VSC on detected tracks

For the first step, calibration target is used. The target is an object with a printed pattern (dots or checkerboard) of known positions on the surface(s). It helps to establish a correspondence between the

2D positions and 3D positions. With cameras covering the entire perimeter of the test section, there will always be some cameras that have to look at the calibration pattern from the back. For our experiments, a transparent calibration target was used. A dot array with a spacing of 8 mm in both x and z direction is laser etched on one side of the surface. The target is turned until only one camera views the target from behind. The refraction index of the material used for the target is different from that of water and thus causes calibration error. As shown in figure 5, although the blue points printed on the target have known 3D positions. They are not used for calibration; the particle positions that actually used for calibration are those "apparent points", where the extension of the light paths from the camera and target surface intersect, as shown by the red symbols. After the initial calibration, a code was used to correct the camera parameters based on the apparent points.

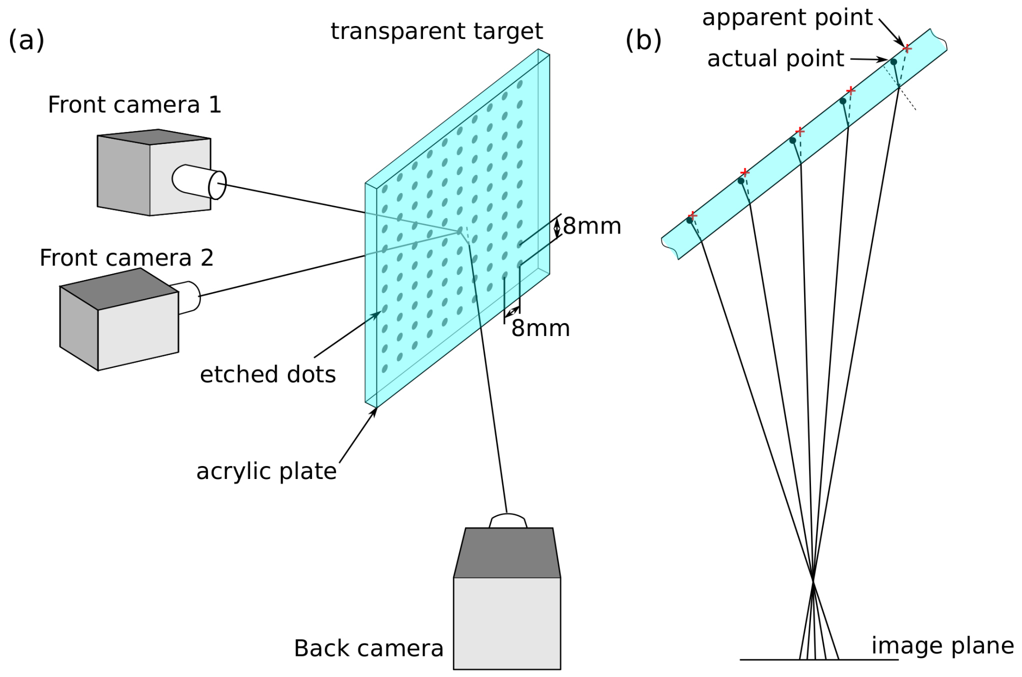

Figure 5: (a) the calibration configuration with two cameras looking at the printed dots from front and one camera viewing the pattern from behind through the transparent target (b) the difference between apparent points and actual calibration dots viewed by the back camera due to the mismatched refractive indices between the target and surrounding water.

Tsai's model (Tsai, 1987) was used to obtain the camera transfer function in terms of camera parameters, including interior parameters, exterior parameters, and distortion coefficients. The least-square fit and nonlinear search algorithm were used to optimize the parameters on the plane where the calibration target was located. Off-plane points could have larger uncertainties and have to be optimized in a different way by using VSC.

Our VSC code is a non-linear optimization code to search for a set of camera parameters that help to minimize the triangulation error of all particles. To find particles with accurate positions, we require that (i) the particle image density is so low that there is no ambiguity and all particles reconstructed should be real even with a large search radius; (ii) the particles selected have to be relatively uniformly distributed throughout the entire view volume; and (iii) only particles that can be tracked for a long distance will be used. After conducting VSC on the low-image-density data, the parameters should be accurate enough to process the high-image-density data. VSC can be run one more time on the high-image-density data to further improve the camera parameters.

Figure 6 shows the PDF of the triangulation error ε before VSC and after two-rounds of VSC. The error systematically decreases from 0.0218 mm to 0.0186 mm after the low-image-density VSC, and continues to reduce to 0.0141 mm after the high-image-density VSC. The results show a significant improvement, and the level of improvement depends on the initial calibration and camera configuration.

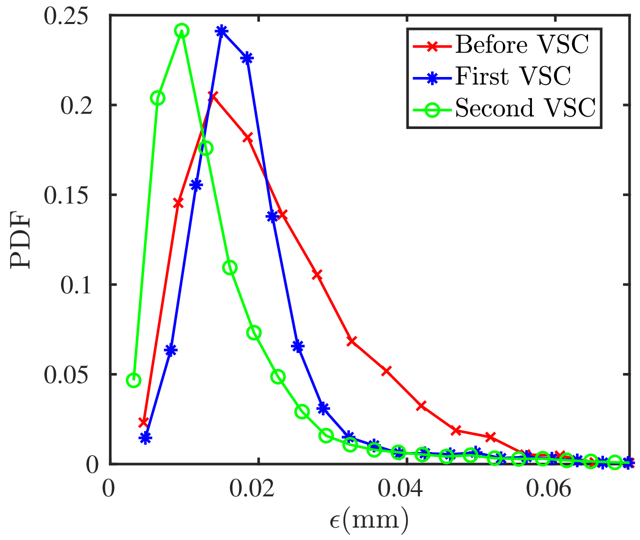

Figure 6: PDF of the triangulation error ε of particles after running VSC on the first low-image-density data and second high-image-density data

6 Experiment result

An experiment was carried out in the vertical water tunnel according to the experiment procedure described in the section 4 and the data was processed with the open-source STB. The mean particle image density is 0.0195 ppp with total of 9,600 particles in the view. Note that the total number of particles seem to be lower than the one reported by Schanz et al. (2016). However, this dataset is imaged by using the back lighting, which is subjected to image noise due to defocused particles outside the view volume. Trajectories from 50 frames are plotted in figure 7 with color showing the magnitude of velocity. Each trajectory was filtered by a Gaussian function to acquire smoothed position, velocity, and acceleration. The difference between the smoothed and raw positions is used to quantify the trajectory quality. For this test dataset, the averaged difference is around 0.01 mm, which is similar to the most probable triangulation error as shown in figure 6.

For one experiment, the camera memory can store 20,000 frames. For four high-speed cameras, it took about 4 seconds of data acquisition. This much data takes 30 minutes to transfer to a local hard drive and another three hours to upload to MARCC. The processing time of one such dataset on MARCC is about one day if we run 10 jobs at the same time. If the calculation was directly performed on the desktop that took the data, the total run time would be around four days, and that is only for one dataset taken within 4 s. This demonstrates the importance of performing data processing on a HPC. Another interesting finding is that the high-image-density datasets also make the postprocessing slow, especially for statistics that need to take a pair of particles. So it makes sense to move the data analysis to HPC to make the entire process manageable.

7 Conclustion

An open-source STB was developed and evaluated with three sets of synthetic images at the image density of 0.0125, 0.025 and 0.05 ppp. The results were evaluated with four metrics, including coverage, matching error, fragmentation index, and correctness. The evaluation shows that the open-source STB performs similarly to the original STB reported by Schanz et al. (2016). The coverage can reach 0.999 even at the image density of 0.05 ppp. Both the fragmentation index and correctness also show that the open-source STB is robust and produce almost perfect results for a noise-free synthetic dataset.

The advantage of the open-source STB is that it is parallelized and can be executed on a high-performance computing cluster. The resources on HPC enables running multiple data at the same time. Careful algorithm efficiency analysis and test were conducted to evaluate the time consumed in each step. The processing time on a desktop of the open-source STB is 40% faster than the reported value on a high-end server back

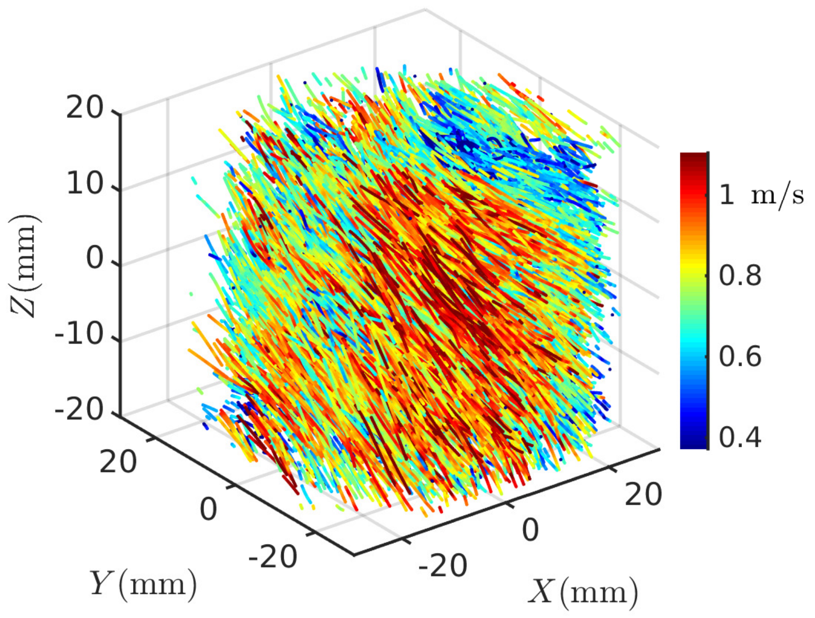

Figure 7: Sample trajectories from one experimental data at the image density of 0.0195 ppp

in 2016. When running on a HPC, the processing are a factor of four faster. This number can easily be increased by running more jobs at the same time.

An in-house VSC code is also implemented for STB. The code is designed for improving camera calibration. In contrast to the VSC reported before, the designed VSC needs to run on a low-image-density data before the high-image-density data to ensure an improved result.

Finally, the code was tested on an experimental dataset with 0.0195 ppp particle image density. The code is now being used for multiple projects and it helps us to process almost all our datasets. We believe that it is at the right time to share the code with the rest of the community. We hope that this open-source STB, once finalized and uploaded to GitHub, will be of value to the community for algorithm development, data assimilation, and uncertainty analysis.

Acknowledgements

This study was financially supported by National Science Foundation under the award numbers: 1705246 and CAREER-1653389.

References

Adamczyk A and Rimai L (1988) Reconstruction of a 3-Dimensional flow field from orthogonal views of seed track video images. *Experiments in Fluids* 6:380–386

Adrian RJ (1984) Scattering particle characteristics and their effect on pulsed laser measurements of fluid flow: speckle velocimetry vs particle image velocimetry. *Applied optics* 23:1690–1691

Chang TP, Wilcox NA, and Tatterson GB (1984) Application of image processing to the analysis of three-dimensional flow fields. *Optical Engineering* 23:283–1

Chiu WC and Rib LN (1956) The rate of dissipation of energy and the energy spectrum in a low-speed turbulent jet. *Eos, Transactions American Geophysical Union* 37:13–26

Elsinga GE, Scarano F, Wieneke B, and van Oudheusden BW (2006) Tomographic particle image velocimetry. *Experiments in fluids* 41:933–947

Hoyer K, Holzner M, Lüthi B, Guala M, Liberzon A, and Kinzelbach W (2005) 3D scanning particle tracking velocimetry. *Experiments in Fluids* 39:923

Kähler CJ, Astarita T, Vlachos PP, Sakakibara J, Hain R, Discetti S, La Foy R, and Cierpka C (2016) Main results of the 4th international PIV challenge. *Experiments in Fluids* 57:97

Li Y, Perlman E, Wan M, Yang Y, Meneveau C, Burns R, Chen S, Szalay A, and Eyink G (2008) A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence* page N31

Maas H, Gruen A, and Papantoniou D (1993) Particle tracking velocimetry in three-dimensional flows. *Experiments in fluids* 15:133–146

Malik N, Dracos T, and Papantoniou D (1993) Particle tracking velocimetry in three-dimensional flows. *Experiments in Fluids* 15:279–294

Ni R, Huang SD, and Xia KQ (2011) Local energy dissipation rate balances local heat flux in the center of turbulent thermal convection. *Physical review letters* 107:174503

Ni R, Kramel S, Ouellette NT, and Voth GA (2015a) Measurements of the coupling between the tumbling of rods and the velocity gradient tensor in turbulence. *Journal of Fluid Mechanics* 766:202–225

Ni R, Puckett JG, Dufresne ER, and Ouellette NT (2015b) Intrinsic fluctuations and driven response of insect swarms. *Physical review letters* 115:118104

Nishino K, Kasagi N, and Hirata M (1989) Three-dimensional particle tracking velocimetry based on automated digital image processing. *Journal of fluids engineering* 111:384–391

Novara M, Schanz D, Gesemann S, Lynch K, and Schröder A (2016a) Lagrangian 3D particle tracking for multi-pulse systems: performance assessment and application of shake-the-box. in *18th international symposium on applications of laser techniques to fluid mechanics*. pages 4–7

Novara M, Schanz D, Kähler C, and Schröder A (2015) Shake-the-box for multi-pulse tomographic systems: towards high seeding density particle tracking in high speed flows. in *11th international symposium on PIV–IV15. Santa Barbara, USA*. pages 14–16

Novara M, Schanz D, Reuther N, Kähler CJ, and Schröder A (2016b) Lagrangian 3D particle tracking in high-speed flows: Shake-the-box for multi-pulse systems. *Experiments in Fluids* 57:128

Ouellette NT, Xu H, and Bodenschatz E (2006) A quantitative study of three-dimensional lagrangian particle tracking algorithms. *Experiments in Fluids* 40:301–313

Papantoniou D and Dracos T (1989) Analyzing 3-d turbulent motions in open channel flow by use of stereoscopy and particle tracking. in HH Fernholz and HE Fiedler, editors, *Advances in Turbulence 2*. pages 278–285. Springer Berlin Heidelberg, Berlin, Heidelberg

Proakis JG and Manolakis D (2007) Digital signal processing, 4th ed

Schanz D, Gesemann S, and Schröder A (2016) Shake-the-box: Lagrangian particle tracking at high particle image densities. *Experiments in fluids* 57:70

Schanz D, Gesemann S, Schröder A, Wieneke B, and Novara M (2012) Non-uniform optical transfer functions in particle imaging: calibration and application to tomographic reconstruction. *Measurement Science and Technology* 24:024009

Schlueter-Kuck KL and Dabiri JO (2017) Coherent structure colouring: identification of coherent structures from sparse data using graph theory. *Journal of Fluid Mechanics* 811:468–486

Schneiders JF and Scarano F (2016) Dense velocity reconstruction from tomographic PTV with material derivatives. *Experiments in fluids* 57:139

Schneiders JF, Scarano F, and Elsinga GE (2017) Resolving vorticity and dissipation in a turbulent boundary layer by tomographic ptv and vic+. *Experiments in Fluids* 58:27

Shen J and Ni R (2017) Experimental investigation of clogging dynamics in homogeneous porous medium. *Water Resources Research* 53:1879–1890

Sheu YH, Chang T, Tatterson GB, and Dickey D (1982) A three-dimensional measurement technique for turbulent flows. *Chemical Engineering Communications* 17:67–83

Tsai R (1987) A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation* 3:323–344

Van Gent P, Michaelis D, Van Oudheusden B, Weiss PÉ, de Kat R, Laskari A, Jeon YJ, David L, Schanz D, Huhn F et al. (2017) Comparative assessment of pressure field reconstructions from particle image velocimetry measurements and lagrangian particle tracking. *Experiments in Fluids* 58:33

Wieneke B (2008) Volume self-calibration for 3D particle image velocimetry. *Experiments in fluids* 45:549–556

Wieneke B (2012) Iterative reconstruction of volumetric particle distribution. *Measurement Science and Technology* 24:024008